Rheinische
Friedrich-Wilhelms-
Universität Bonn

Institute for Computer Science
Department VI
Autonomous Intelligent Systems

Rheinische
Friedrich-Wilhelms-Universität Bonn

Master Thesis

Object Localization for Warehouse Automation

Author:
Max Schwarz

First Examiner:
Prof. Dr. Sven Behnke

Second Examiner:
Prof. Dr. Christian Bauckhage

Date: February 4, 2017

Declaration

I hereby declare that I am the sole author of this thesis and that none other than the specified sources and aids have been used. Passages and figures quoted from other works have been marked with appropriate mention of the source.

Place, Date

Signature

Abstract

Object detection is an important capability for manipulation robots, enabling search and retrieval of known objects. Particularly in warehouse automation contexts, object detection is the basis of picking automation.

This thesis aims to provide an end-to-end bin-picking pipeline including object detection, grasp selection, and motion execution, focusing on the object detection part. As a basis, a state-of-the-art neural network for dense captioning is adapted to the object detection task. Due to the recent good availability of RGB-D sensors, part of the thesis focuses on RGB-D stream fusion from two cameras and incorporation of depth measurements into the object detection pipeline.

The design choices made for the object detector are evaluated and confirmed on a custom warehouse dataset. Additionally, the method is evaluated on a disaster response dataset to show general applicability. Using a simple pixel-wise product, object detection output can improve results from external semantic segmentation.

The object detector is integrated into a complete system for the Amazon Picking Challenge 2016. To facilitate picking, a custom inverse kinematics solver is developed for the special constrained situation of reaching into a warehouse shelf. Grasps are selected heuristically using the perception results. Motions are generated using parametrized motion primitives. Finally, the entire robotic system was evaluated at the Amazon Picking Challenge 2016, where it reached a second and third place.

Contents

1. Introduction

With robots employed in more and more tasks, it becomes clear that perception is one of the main challenges faced by robotic systems. While there are robots that can move with extreme precision and speed, human perceptive capabilities are still unmatched.

A key problem in robotic perception is object detection, which attempts to solve both classification (which object are we seeing?) and localization (where is the object?) simultaneously. This capability is needed in many applications, mostly whenever items need to be found or retrieved. Examples include warehouse applications, home assistance robotics, rescue robotics, and many others. Of course, there are also non-robotic applications of object detection, such as image mining and automatic image annotation.

This thesis strives to approach object detection from a strongly application-oriented perspective. While the developed method is generally applicable, many design choices were motivated by the Amazon Picking Challenge 2016 (APC 2016), a robotics challenge organized by Amazon focusing on warehouse automation. Specifically, contestants were required to build robotic systems capable of retrieving items from unordered shelf storage, and storing items into the shelves.

In recent years, depth sensors have become standard in robotics applications. Since common object detection methods target the color modality only, this thesis also investigates methods to incorporate depth measurements into the object detection pipeline.

After defining the object detection problem and corresponding metrics in Chapter 2, discussing related work in Chapter 3, and describing the robotic system in Chapter 4, the thesis will contain the following contributions:

1. Adaption of a state-of-the-art object detection architecture to a warehouse automation situation, including the incorporation of depth measurements (Chapter 5),

2. a component-level evaluation of said method and the involved design decisions on two datasets (Chapter 6),

3. integration of the method into the robotic system, including heuristic grasp

selection, intelligent inverse kinematics, and motion generation using motion primitives (Chapter 7), and

4. a system-level evaluation at the APC 2016 (Chapter 7).

Finally, Chapter 8 offers a conclusion and points out possible extension points for future work.

2. The Object Detection Problem

2.1. Definition

The object detection problem is posed as follows: For an input image I, the detector should output n detections D_i, where n is the number of *relevant* items present in the image. Each detection record should contain a precise localization of the corresponding item. Usually, the object detection problem is formulated for 2D color images, and the detections are output as 2D axis-aligned bounding boxes (see Figure 2.1).

Detectors usually output a confidence score c_i for each detection D_i. This allows to rank all detections for one class predicted in one image, or even over multiple images. In the context of bin picking, this score can be used to determine the most reliable detection to use for retrieval.

In contrast to image classification, where each image has one ground truth label, measuring detector performance is more involved. On the one hand, one wants a reliable confidence score, so that the detector output is trustworthy, on the other hand, one desires a precise localization of the items.

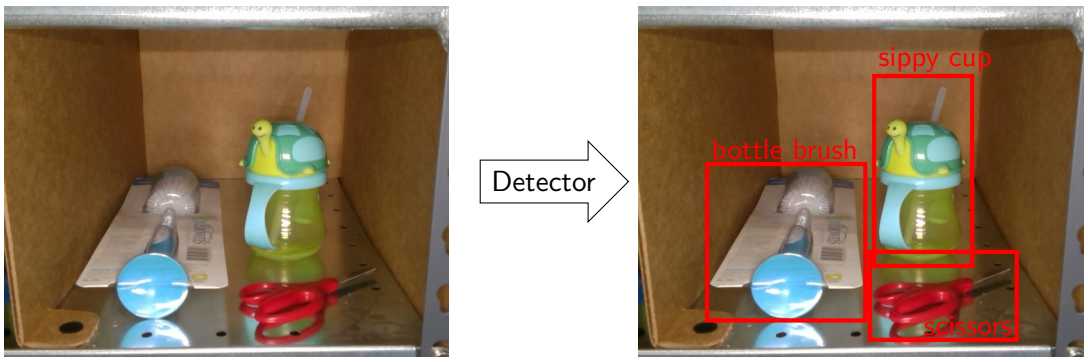


Figure 2.1: Typical 2D object detection. An input RGB image is processed and three detections with 2D axis-aligned bounding boxes are predicted.

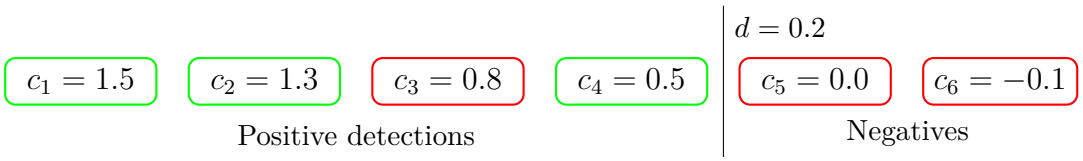


Figure 2.2: Output of a ranking detector. The detections are ordered with descending confidence c_i from left to right. The border color denotes the ground truth label (green: positive example, red: negative). The threshold $d = 0.2$ allows to obtain binary classification output.

2.2. Evaluation Metrics

2.2.1. mAP Metric

The most popular metric for evaluating object detection approaches is the *mean Average Precision (mAP)* metric. This metric is based on the notions of *precision* and *recall* from binary classification:

$$precision = \frac{tp}{tp + fp}, \text{ and} \tag{2.1}$$

$$recall = \frac{tp}{tp + fn}, \tag{2.2}$$

where tp denotes the number of true positives (i.e. correctly classified positive examples), fp the number of false positives (i.e. wrongly classified negative examples), and fn the number of false negatives (i.e. wrongly classified positive examples). In other words, precision represents the fraction of relevant detections in the algorithm output, while recall measures how many of the underlying relevant items are detected.

If a confidence score c_i is available for each detection D_i, we can turn such a detector into a binary detector using a threshold d: A detection i is declared positive if and only if $c_i \geq d$ (see Figure 2.2). Note that d is a parameter of this detector. If we want to evaluate the performance, we can plot precision and recall for each value of d, a so-called precision-recall curve (see Figure 2.3a). Figure 2.3b) and c) show real results from our APC dataset described in Chapter 7. Average Precision (AP) is then simply the average precision value in this plot. Since object detectors output detection confidences for each object class, the AP values are calculated independently for each class and then averaged to obtain the mean AP (mAP).

In the general situation, the shape of a precision-recall curve is entirely unconstrained. For the aforementioned thresholding construction, recall increases

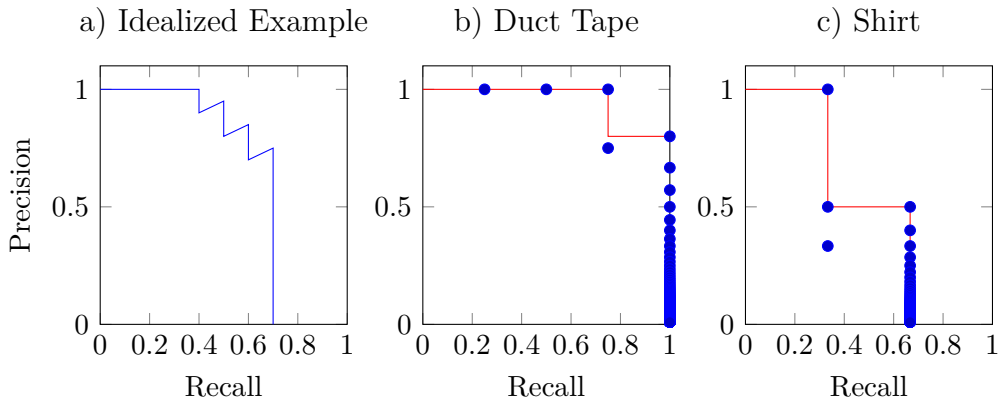


Figure 2.3: Example precision-recall curves. (a) Usual example of a precision-recall curve. (b) and (c) show real precision-recall points (blue) on one split of the APC dataset, see Chapter 7. Note that in (a) and (c) the detector never finds all positive examples, so that recall never reaches one. (b) and (c) exhibit one and two false positive(s) ranked before the last true positive detection, respectively. The effect of maximum interpolation is shown in red.

(weakly) with decreasing d. Precision, however, may decrease when there is a new false positive "uncovered" by the threshold, but it may also increase again if there are new true positives which counter already present false positives. Near-optimal algorithms will always result in near-monotonic curves, however, as they rank the positive examples higher than the negatives.

Many researchers, whole datasets, and competitions (e.g. PASCAL VOC since 2007, Everingham et al. 2010) use a max-interpolated AP, i.e. force the curve to be monotonically decreasing by means of a right-looking maximum operation. Formally,

$$AP_{\max}(p) = \max_{p \le p' \le 1} AP(p'). \tag{2.3}$$

The motivation seems to be that one could have chosen the looser d threshold obtaining the maximum instead of using the current sub-optimal d value (see red mark in Figure 2.3b-c). However, this motivation seems weak, since the AP metric strives to give the average algorithm precision over *all* choices of d. Nevertheless, since interpolated average precision is so widely used, it is also adopted in this thesis to allow for easy comparison.

In the context of object detection, the notion of a true positive has to be defined. Usually a detection D is counted as a true positive whenever its Intersection over Union

$$\mathrm{IoU}(D, G) = \frac{|D \cap G|}{|D \cup G|} \tag{2.4}$$

with the closest ground truth rectangle G is greater than 0.5 (Everingham et al. 2010). Note that detections are associated to ground truth rectangles greedily with decreasing confidence, i.e. a well localized detection will be hidden by a badly localized detection with higher confidence. This means that object detectors have to suppress multiple positives somehow, e.g. using greedy non-maximum suppression.

In this thesis, two mAP values will be reported: *Uninformed* mAP and *Informed* mAP. In the informed case, detections are masked by the information which items are present in each frame, discarding detections of objects not present. This is particularly useful in a warehouse automation scenario, since there usually is a warehouse database which can supply this information.

2.2.2. F1 Metric

Generally, the mAP metric as defined above places greater weight on correct detection than on precise localization. Indeed, it is possible to achieve perfect average precision scores (and we will see such results in Chapter 7), while achieving perfect localization precision is much harder.

To also provide sensitivity in this direction, one can define a metric for object detection based on pixel-level precision and recall. In this case, we consider the particular use case for an object detector in the context of warehouse automation: It is known that a particular object resides in a particular shelf bin, and we need to retrieve it. Here, we are only interested in the detection i with maximum confidence c_i for this object class. We measure its precision and recall:

$$precision = \frac{|D \cap G|}{|D \cup G|} = \mathrm{IoU}(D, G) \tag{2.5}$$

$$recall = \frac{|D \cap G|}{|G|}, \tag{2.6}$$

where D is the detected bounding box and G denotes the closest ground truth bounding box. Note that a complete mislocalization results in zero precision and recall.

For evaluation on a test set, the F1 scores are computed for each ground truth label, and then averaged for each class. It should be noted that this metric does not punish false positives for classes not present in the ground truth for a particular image. Such behavior is better quantified by the mAP metric.

3. Related Work

3.1. Robotic Bin Picking

The bin picking problem, i.e. retrieving items from a more or less unordered storage system, is hard. It combines perceptual problems (where are the objects?) with planning problems (how to get them out?) and mechanical problems (how can we pick the objects?). On the other hand, there is a high demand for solutions in the industry. Therefore it is not surprising that there is a long line of research works in this area.

Usually, the works focus on the perception side. Drost et al. (2010) propose a method for creating a global model based on point pair features and register it efficiently with 3D point cloud scenes. The sparse model and scene point clouds result in very high performance. However, the method is limited to geometry and is targeted towards single-class situations. Nieuwenhuisen et al. (2013) detect objects using shape primitives. Primitives are automatically detected in both the object and scene point cloud. Using a graph matching method, correspondences in the neighboring graph of primitives are found. The authors also provide solutions for global navigation to the bin picking area, active perception for avoiding occlusions, and finally grasp planning using a multiresolution height map. The approach is again limited to geometry and rigid objects which can be modeled using the available shape primitives. Berner et al. (2013) enhance the method using 2D primitives such as circular contours. Pretto, Tonello, and Menegatti (2013) describe an object perception system for planar objects from monocular color images. Domae et al. (2014) focus on grasp planning in depth images for random objects. The method does not use object models, instead it tries to find graspable spots in the observed geometry. Buchholz et al. (2014) also find grasp poses on unknown objects, but can perform pose estimation for precise placement using an inertial model during the grasp movement. The needed measurement and computation time is thus hidden in the robot motion. Kaipa et al. (2016) address the different failure modes that can happen during bin-picking, such as misrecognitions, failed grasps, placement inaccuracies, and others. If possible, the failure is automatically detected and corrected, or a human supervisor is alerted and can solve the problem using a remote interface.

The Amazon Picking Challenge 2015 resulted in various approaches for a very general shelf-picking problem. Correll et al. (2016) aggregate lessons learned during the APC 2015 and present a general overview and statistics of the approaches.

Eppner et al. (2016) describe their winning entry for the APC 2015. A combination of a mobile base with a 7-DOF arm can reach all shelf bins comfortably. In contrast, the robotic system used in this thesis employs a large enough arm that the robot base can be fixed in position (see Section 4.1). Eppner et al. (2016) use a fixed suction gripper, which can execute top and side picks. Suctioning an object from the front is not possible. Object perception is performed using a wrist-mounted RGB-D sensor. Six hand-crafted pixel-wise features are computed, including color and geometry-based features. Histogram backprojection is used to estimate posterior probabilities for each object class. The target object is found by searching for the pixel with maximum probability. Top or side grasps are selected heuristically using the extracted 3D bounding box. Similar to the approach described in Section 7.1, motions are generated from parametrized motion primitives and possible collisions are detected using feedback from the robot arm. The team scored 148 out of 190 possible points, resulting in the first place.

The second place went to Yu et al. (2016) with Team MIT. In contrast to Eppner et al. (2016) and the approach in Section 4.1, their design includes a hybrid endeffector which can grip objects as well as apply suction. An industrial arm provides high accuracy and speed. Here, object perception is also based on RGB-D measurements from a wrist-mounted sensor. A depth-only GPU-based instance recognition approach is used to determine object poses. Again, motion primitives are executed to pick the items. As a specialty, there are motion primitives which only intend to change the configuration inside the shelf bin—such as tipping an object over on its side, so that it is more accessible.

The 2016 Amazon Picking Challenge included more difficult objects: Heavy objects such as the 3 lb dumbbell, transparent or reflective objects (water bottle), and highly non-rigid objects (shirt). Also, the general difficulty in the arrangements and the number of items increased. Finally, the new stowing task was introduced.

Hernandez et al. (2016) reached first place in both the picking and the stowing task in the APC 2016. Their system consists of a large industrial 7-DOF arm mounted on a horizontal rail, resulting in eight degrees of freedom in the arm and base. The gripper is a complex custom design, allowing both suction and pinch grasps. Like the design described in Section 4.1, the suction cup can be bent to facilitate top, side, and frontal grasps. Object perception is based on RGB-D measurements from an Ensenso 3D camera. The authors report problems with reflections and noise, and therefore built in heuristics to reject false registrations. In contrast, we added a second RGB-D camera to be able to filter out false measure-

ments (see Section 4.2). Similarly to our architecture, object detection is carried out using an approach based on Faster R-CNN (Ren et al. 2015). After detection, object poses are estimated using a point cloud registration method. For grasp planning, primitive shapes are fitted to find grasp candidate spots. Candidates are then filtered using reachability measures. For deformable objects, a simple measurement-based heuristic is used like in Section 7.1.1. It seems that planned motion trajectories were executed mostly without feedback, so motion planning had to be employed to be sure that the trajectories are collision-free. In contrast, the inverse kinematics presented in Section 7.1.2 prevents collisions with the static environment directly in the IK solver, and motion execution stops as soon as unexpected contact forces occur, allowing the system to continue with another item, retrying the failed item at the end.

3.2. Object Detection

The first object detection framework to offer robust detection rates in real-time was proposed by Viola and Jones (2001), who train Haar cascades for this task. While mainly targeted and used for face detection, the method is actually applicable for general objects.

Recently, deep learning techniques have become state-of-the-art in many computer science disciplines, including machine translation (Sutskever, Vinyals, and Le 2014), speech recognition (Graves, Mohamed, and Hinton 2013), and computer vision (Krizhevsky, Sutskever, and Hinton 2012). This trend is powered by the availability of massively parallel computing architectures (e.g. GPUs) and the large amounts of available labeled datasets (Krishna et al. 2016; Russakovsky et al. 2015; Song, Lichtenberg, and Xiao 2015). In the context of computer vision, deep learning approaches excel in image classification (Krizhevsky, Sutskever, and Hinton 2012), object detection (Johnson, Karpathy, and Fei-Fei 2016; Liu et al. 2016), and semantic segmentation (Chen et al. 2015; Long, Shelhamer, and Darrell 2015).

For object detection, one line of work considers the task as region proposal followed by classification and scoring. Girshick et al. (2014) process external region proposals using Region of Interest (RoI) pooling to reshape intermediate CNN feature maps to a fixed size. To increase performance, all regions may be processed in a single forward pass (Girshick 2015). Finally, region proposal networks that regress from anchors to regions of interest are integrated (Ren et al. 2015). Note that the RoI pooling layer is not easily differentiable, so the methods usually use an alternating training scheme for the parts before (RPN) and after the layer

(classification). The object detection approach described in Chapter 5 is based on the DenseCap region detection and captioning approach of Johnson, Karpathy, and Fei-Fei (2016), which builds upon the ideas of this line of research.

Another line of work performs bounding box regression and classification jointly for a fixed set of anchor locations in the image. Redmon et al. (2015, "YOLO") introduce this idea and obtain very good generalization capabilities of their models. Liu et al. (2016, "SSD") obtain slightly better detector performance on the Pascal VOC dataset. The advantage of these methods is their very fast computation—since the models are fully convolutional and avoid costly recomputations for each bounding box, prediction times are in the order of 20 ms. The lower computation load also means that these networks can more easily be evaluated on low-power embedded hardware—which is of particular interest for mobile robots. On the other hand, proposal and classification are more tightly coupled, which can limit extensibility (see Chapter 5).

In the context of RGB-D object detection, the main problem discussed in the community is the lack of labeled depth (or even RGB-D) training data sufficient for supervised training. Hoffman et al. (2016) propose a mid-level fusion method that is able to learn on RGB-D data with partially available depth. The method can transfer learned knowledge from a category with depth available to a category where depth data is only available at test time. Gupta, Hoffman, and Malik (2015) introduce a distillation scheme, where a reference RGB CNN is used to generate supervision data in order to train a depth CNN on unannotated RGB-D frames. This method is also evaluated in this thesis as one of the methods for incorporating depth (see Sections 5.4.4 and 6.1.1).

4. Robotic System for the APC 2016

Figure 4.1: Team NimbRo's robot at the APC 2016, showing the entire work cell layout with shelf, tote, and robot.

The main application of the methods described in this thesis was the Amazon Picking Challenge 2016[1], which took place June 30th to July 3rd, 2016 in Leipzig, parallel to RoboCup 2016. In addition to picking objects from Amazon shelves, which had been required in the Picking Challenge 2015, a second competition focused on stowing objects from totes back into the shelves. Figure 4.1 shows such a shelf and tote, with our robot in front of it.

The 39 available items (see Figure 4.2) varied strongly in shape, weight, texture, and other properties. In particular, there were very heavy objects (e.g. a 3 lb dumbbell), highly non-rigid objects (a T-shirt), very small objects (scissors), reflective objects (metal dog bowl), large objects (paper towels), transparent objects

[1] `http://web.archive.org/web/20160903010523/http://amazonpickingchallenge.org/`, original unavailable.

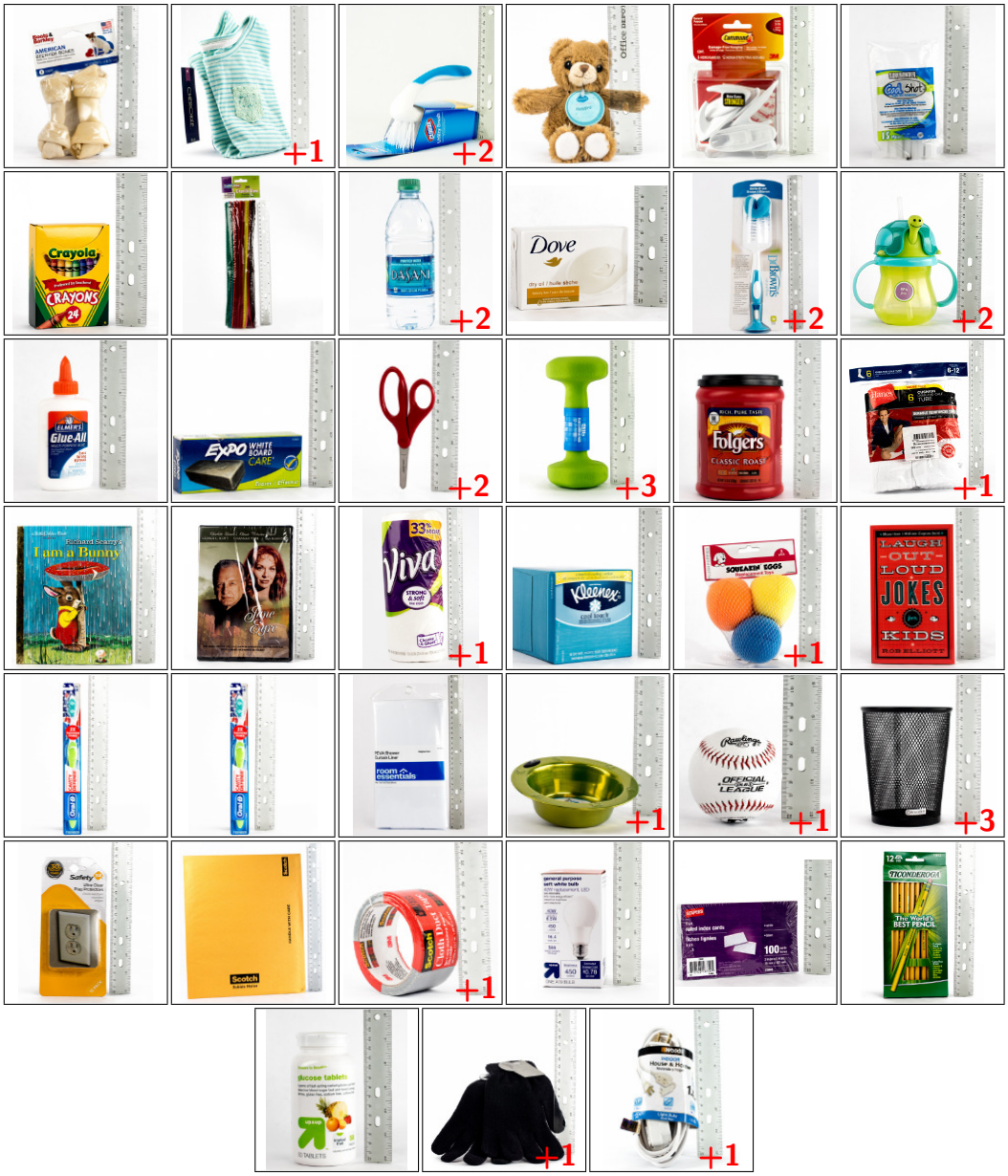


Figure 4.2: The 39 different APC objects. Bonus points for hard objects are shown in red. Individual images provided by Amazon.

(water bottle), and objects which were hard to pick using suction (meshed pencil cup). Some of these "hard" objects were attributed with bonus points for successful picking (see Figure 4.2). To facilitate training, the objects were distributed to the participants prior to the competition.

For each task, 12 items had to be moved into or out of the shelf in 15 minutes, as dictated by a job file, which was handed out just before the competition run. The system was required to output the location of all objects after the run (e.g. for updating a warehouse management system database).

The fixed structure of the shelves imposed hard constraints on the robotic systems, in particular the shelf bins themselves were very narrow and deep. Additionally, the bin floors were made from highly reflective metal, which caused mirroring effects for the perception system. Note that both color and depth modalities were affected by the mirror images, since the metal was also reflective in the infrared spectrum.

Please note that many contributions by other team members in the areas of high-level control, low-level actuator and sensor interfacing, semantic segmentation, object registration, and mechanical design were necessary to allow the system to succeed. The entire system is described in (Schwarz, Milan, et al. 2017).

4.1. Mechanical Design

While mechanical design is not the focus of this thesis, a short description of the robot is nevertheless necessary for understanding later contributions. To avoid designing the entire system from scratch, an off-the-shelf robotic arm was chosen, the Universal Robots UR10. The UR10 workspace is large enough to cover movement in front of all bins of the shelf. Additionally, the UR10 is cost-effective, lightweight, and offers safety features such as an automatic (and reversible) stop upon contact with the environment.

To reach into the shelf, we added an extension with a prismatic joint, capable of extending 37 cm (see Figure 4.3). On the tip of the extension, a rotary joint enables the robot to approach objects from the front, top, or from the sides. Finally, the gripper is realized as a vacuum gripper - optimized for high vacuum *and* high air flow. The vacuum is generated by a 3100 W vacuum cleaner mounted in the base of the robot. Vacuum strength can be regulated by a motorized bleed valve.

The system uses two dedicated computers. The first one contains an Intel Core i7-4790K (4 GHz) processor and controls the hardware and high-level operations, while the second one contains two Intel Xeon E5-2670 v2 (2.5 GHz) and four NVIDIA Titan X GPUs and is used for vision processing. For training, all four

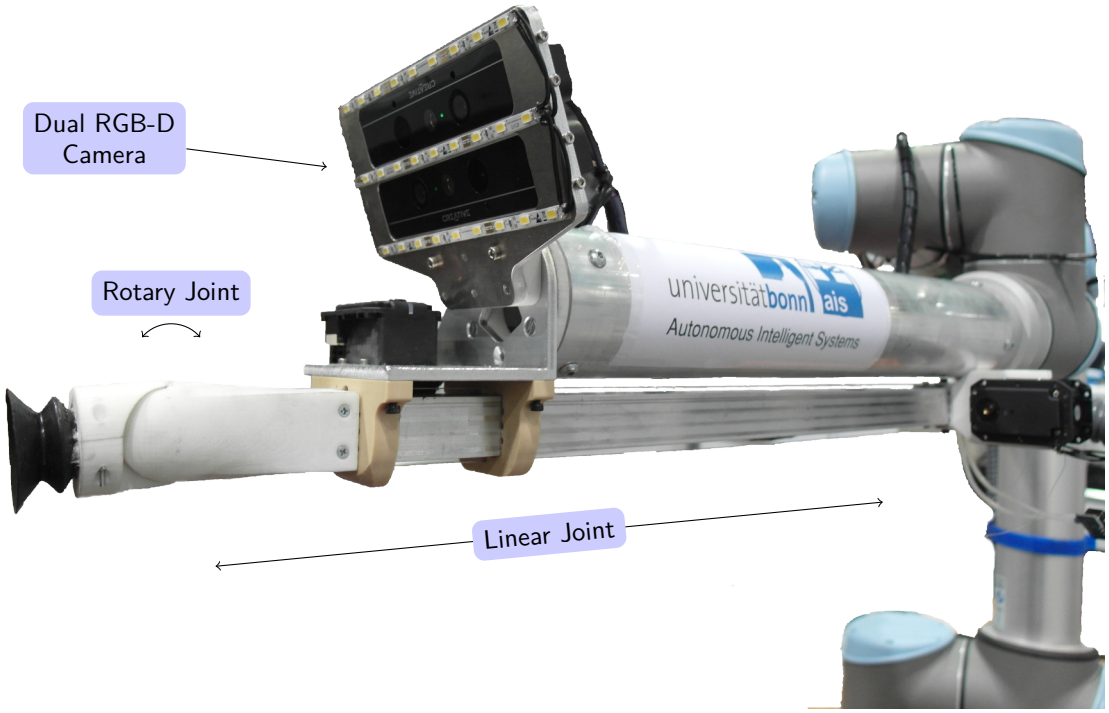
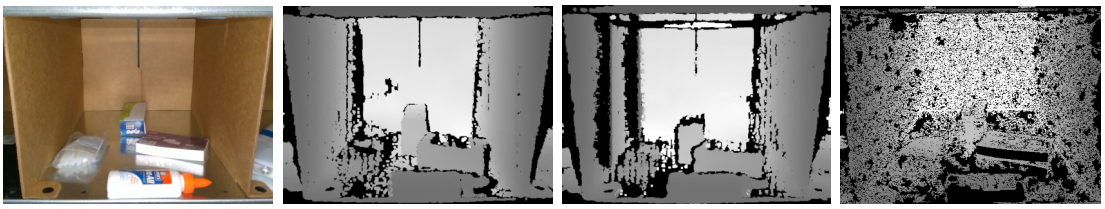


Figure 4.3: Endeffector with suction finger and dual camera setup.

GPUs can be used to accelerate training. At test time, two GPUs are used in parallel for the object detection method described in Chapter 5 and the semantic segmentation method (see Section 5.6).

4.2. RGB-D Capture

Due to its high resolution, low weight, and short minimum sensing range, the Intel RealSense SR300 RGB-D sensor was chosen after experiments with multiple



(a) RGB frame (b) Upper depth (c) Lower depth (d) Stereo depth

Figure 4.4: RGB-D capture from two sensors. Note the corruption in the left wall in the lower depth frame.

sensors. However, the sensor sometimes produces systematic artifacts on the walls of the shelf (see Figure 4.4(c)). Because the artifacts seem to depend on the viewing angle, they appear only on one side of the image, i.e. the left wall from the sensor perspective. To be able to filter out these artifacts, we added a second SR300 sensor rotated by 180° (see Figure 4.3). Image acquisition is done in sequence, so that only one infrared projector is active at a time to prevent interference. Since the scene is static, this poses no problem. To reduce the effect of unknown external illumination, the endeffector also carries LED strips for local independent lighting.

The RGB stream from the secondary sensor can also be exploited for calculating depth from stereo information. This gives a third depth source which can be used for tie-breaking between the two SR300 depth streams. Dense stereo disparity between the two RGB cameras is calculated using LIBELAS (Geiger, Roser, and Urtasun 2010), which was selected for its real-time capability and dense output. Since the implementation and a ROS wrapper is freely available, it was straightforward to integrate into the system.

5. Object Detection Method

5.1. Starting Point: DenseCap

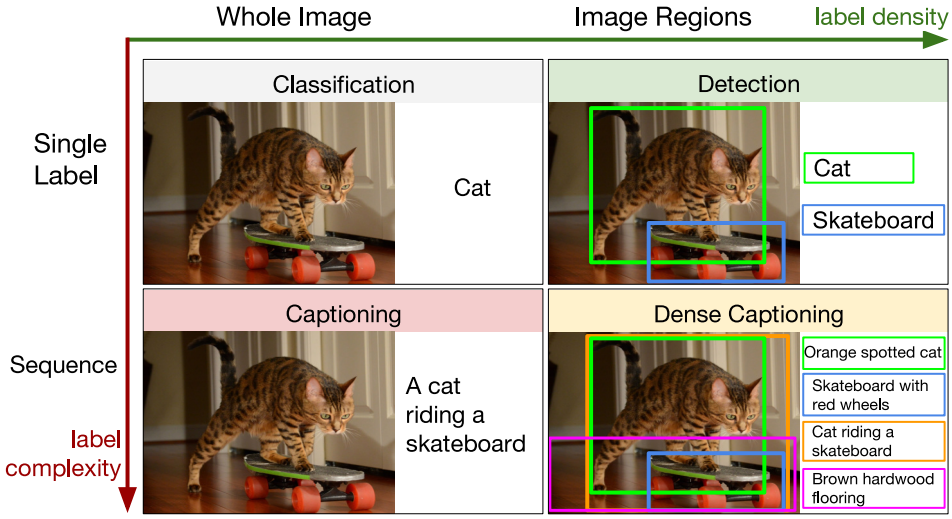


Figure 5.1: Relationship of Object Detection and Dense Captioning. Taken from Johnson, Karpathy, and Fei-Fei (2016).

The object detection method discussed in this thesis is based on the DenseCap network of Johnson, Karpathy, and Fei-Fei (2016). This is somewhat curious, since DenseCap solves another Problem: The Dense Captioning task (see Figure 5.1), whose objective is to find interesting regions (bounding boxes) in an input image and create detailed textual descriptions for each of the regions. However, this goal is closely related to object detection and DenseCap offers unique advantages, as we will see below. Briefly speaking, DenseCap receives an RGB image as input and outputs numerous rectangles in the image, ranked by confidence, each accompanied by a textual description of the rectangle contents.

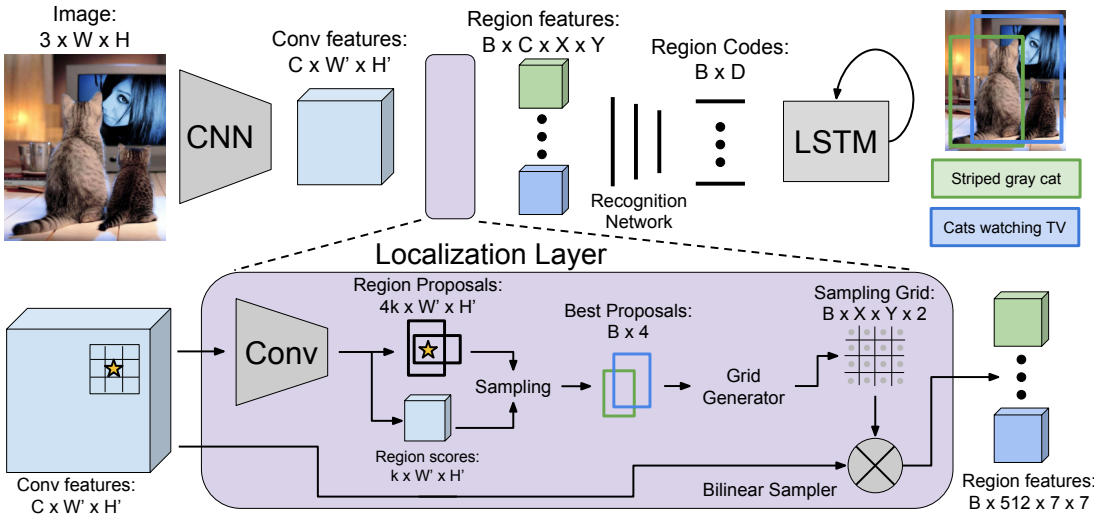


Figure 5.2: Original DenseCap architecture. Taken from Johnson, Karpathy, and Fei-Fei (2016).

Figure 5.2 shows an overview of the DenseCap network. In more detail, the network first extracts features using the convolutional part of the VGG-16 CNN (Simonyan and Zisserman 2014), which was pretrained on ImageNet (Russakovsky et al. 2015) for image classification. A convolutional region proposal network then creates a large number of region proposals from a list of fixed anchor boxes. The proposals are NMS-sampled[1] to a fixed number (1000 proposals) using a confidence score. The intermediate CNN features are then interpolated bilinearly from each proposal region to a fixed size. The fully-connected part of VGG-16 (up until the second-last layer) is then used to compute a high-level feature vector for each proposal region. Finally, an LSTM network generates textual descriptions for the regions. In addition to the pretrained CNN part, the entire pipeline is trained on the Visual Genome dataset (Krishna et al. 2016).

While DenseCap is targeted for the somewhat more complex dense captioning problem, it nevertheless has several advantages over related object detection works, some of which have been stated in Chapter 3, but will be repeated here:

End-to-end training. In contrast to similar architectures for object detection (e.g. Faster R-CNN, Ren et al. 2015), the DenseCap architecture can be trained end-to-end without the need for approximations, since its bilinear interpolation layer is fully differentiable.

External proposals. Since DenseCap includes a general-purpose classification net-

[1]NMS: Non-maximum suppression, usually implemented as sorting by confidence and greedily taking proposals while observing a maximum intersection threshold.

work, it is straightforward to consider external object proposals. This will be exploited later in Chapter 7 to add object proposals from RGB-D cues. Other architectures have classifiers which are strictly tied to fixed anchors in the image and thus cannot easily be used on external proposals (YOLO, Redmon et al. 2015).

Popular framework. DenseCap is based on Torch[2], a popular deep learning framework for the LUA scripting language. Other object detection works are often based on entirely custom frameworks (YOLO, Redmon et al. 2015) or use heavily modified forks of the Caffe framework (Faster R-CNN, Ren et al. 2015). Both make further modifications and combinations with other techniques rather difficult.

5.2. Object Classification

The generated textual descriptions are not needed for object detection. However, the captions are generated from the intermediate high-level VGG-16 feature vectors, which must therefore be highly descriptive. To exploit the power of this feature representation, the network is used up until the LSTM, which is replaced with a classification layer. Two methods for classification are investigated and will be described in the following.

5.2.1. SVM Classification

Linear SVMs excel in learning from few examples. There are many examples for combinations of pretrained CNN feature extractors with a linear SVM for classification, including the author's prior work on object classification and pose estimation (Schwarz, Schulz, and Behnke 2015). In this case it seems natural to combine the pretrained DenseCap architecture with a trained SVM for region classification. The SVM score (the signed distance from the hyperplane) is used as the detector score.

Notably, linear SVM training is very efficient and can be completed in few seconds for cases with ten object classes and even a large number of training examples (around 400,000 including all negative examples). Particularly, the SVM training can be constrained to the classes actually present in the image. In many cases, this information is available (e.g. from a warehouse management system). In this case, SVM training can even be performed on-the-fly just before perception.

[2]http://torch.ch/

(a) RGB frame (b) Upper depth (c) Lower depth (d) Stereo depth (e) Fused result

Figure 5.3: RGB-D fusion from two sensors. Note the corruption in the left wall in the lower depth frame, which is corrected in the fused result.

Especially in warehouse automation contexts, training time is usually bounded, since the number of items in a particular bin is bounded.

5.2.2. Softmax Classification

While the SVM approach is promising, it does not change the region proposals generated by the network. In order to to refine the proposals, the DenseCap pipeline needs to be fine-tuned end-to-end (optionally excluding the initial VGG-CNN parts). This is investigated using a softmax classification layer, which is fully differentiable and thus straightforward to train using backpropagation of error.

5.3. RGB-D Preprocessing

While depth is especially valuable for understanding the geometry of the scene, e.g. for planning an object grasp, it is also desirable to make use of the additional modality for the object detection pipeline. Since color and depth, while closely related, offer complementary information, one can hope that detection performance increases with the added information.

As described in Section 4.2, the two sensors capture two RGB streams, and three depth streams: Two directly measured by the sensors, and stereo disparity computed from the two RGB streams. The three depth streams are projected into one common frame (the upper camera in our case) and are then fused using a linear combination with predefined weights α_i[3]. In particular, the stereo stream is fused using a low weight, since the depth measurements of the SR300 cameras are usually more precise (but not always available). Finally, we distrust measurements where the different depth sources disagree by introducing an additional "spread" weight w. In summary, we obtain the following equations for combining depth

[3]The experiments use $\alpha_{\text{stereo}} = 0.1$ and $\alpha_{\text{RGB-D}} = 40.0$.

measurements D_i of a single pixel to a depth measurement D and weight w:

$$D = \frac{\sum \alpha_i D_i}{\sum \alpha_i}, \tag{5.1}$$

$$w = \exp(-(\max_i D_i - \min_i D_i)) \sum \alpha_i. \tag{5.2}$$

Pixels where $\max_i D_i - \min_i D_i > 5\,\text{cm}$ are disregarded entirely. Figure 5.3 shows an exemplary scene with individual raw sensor measurements as well as the fused depth map.

Since the resulting fused depth map is usually sparse, and nearly all perception modules (object detection, semantic segmentation, grasp selection) require dense depth measurements, a filling operation is needed. We follow the work of Ferstl et al. (2013), who upsample depth images guided by a high-resolution grayscale image. In contrast to many other guided upsampling approaches, this one does not assume a regular upsampling grid. Instead, any binary (or even real-valued) weight matrix can be used to specify the location of source pixels in the output domain. This makes the approach applicable to the present scenario, where the mask of valid pixels has no inherent structure. Invalid pixels are filled using a Total Generalized Variation (TGV) prior, guided by the RGB image. In the experiments, the HSV saturation channel proved better than grayscale intensity for guiding the filling operation.

The upsampling is formulated as an optimization problem, minimizing the energy term

$$\min_{u,v} \left[\alpha_1 \int_{\Omega_H} |T^{\frac{1}{2}}(\Delta u - v)| \,\mathrm{d}x + \alpha_0 \int_{\Omega_H} |\Delta v| \,\mathrm{d}x + \int_{\Omega_H} \mathbf{w} |(u - D_s)|^2 \,\mathrm{d}x \right], \tag{5.3}$$

where the first two summands regularize the solution using Total Generalized Variation, and the last summand is the data term, weighted by the weights w calculated above. For details about the problem formulation and the solver algorithm, we refer the reader to Ferstl et al. (2013). The guided upsampling was implemented in CUDA to achieve near real-time performance (<100 ms per image).

Note that this guided upsampling was implemented after the APC competition. During the competition, we employed a recursive smoothing filter guided by the RGB image. It was implemented using the Domain Transform technique described by Gastal and Oliveira (2011).

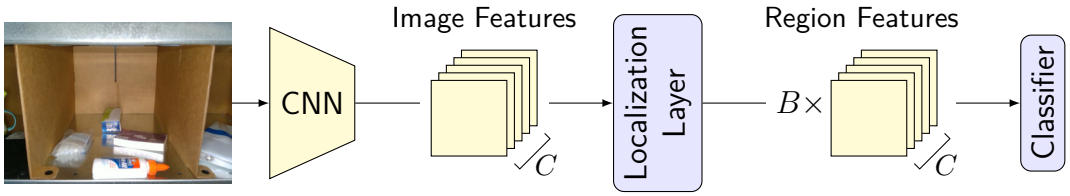


Figure 5.4: Simplified overview of RGB-only feature extraction, region proposal, and classification. C denotes the number of CNN feature maps after the last convolutional layer (512 for VGG-16). The internal proposal generator produces B proposals (1000).

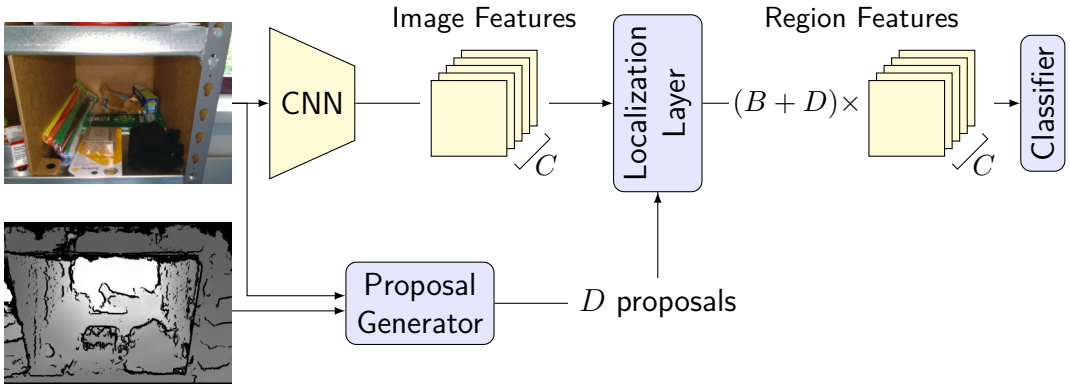


Figure 5.5: Detection pipeline with external proposals. C denotes the number of CNN feature maps after the last convolutional layer (512 for VGG-16). The internal proposal generator produces B proposals (1000). Additional D proposals are added from the external proposal generator based on RGB and depth data.

5.4. Exploiting Depth Measurements

Having a fused depth map available, the question is now how to integrate the depth measurements into the object detection pipeline. Figure 5.4 shows the unmodified object detection pipeline which will be extended with several methods for incorporating depth in the following.

5.4.1. External RGB-D Proposals

Since external proposals can be injected easily into the DenseCap pipeline after the region proposal network, one way to use depth information is to simply feed in RGB-D-based proposals (see Figure 5.5). As one particularly straightforward and fast implementation, a connected component algorithm can be used to generate additional proposals. Two pixels are deemed connected if they do not differ more than a threshold in terms of 3D position (5 mm), normal angle (50°), saturation,

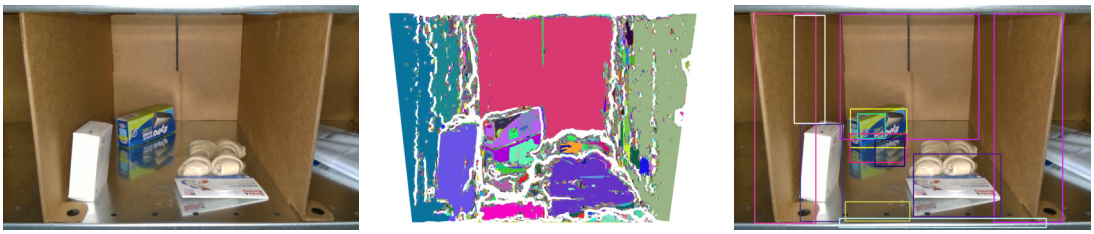


Figure 5.6: RGB-D based additional region proposals. Left: RGB frame. Center: Regions labeled using the connected components algorithm. Right: Extracted bounding box proposals.

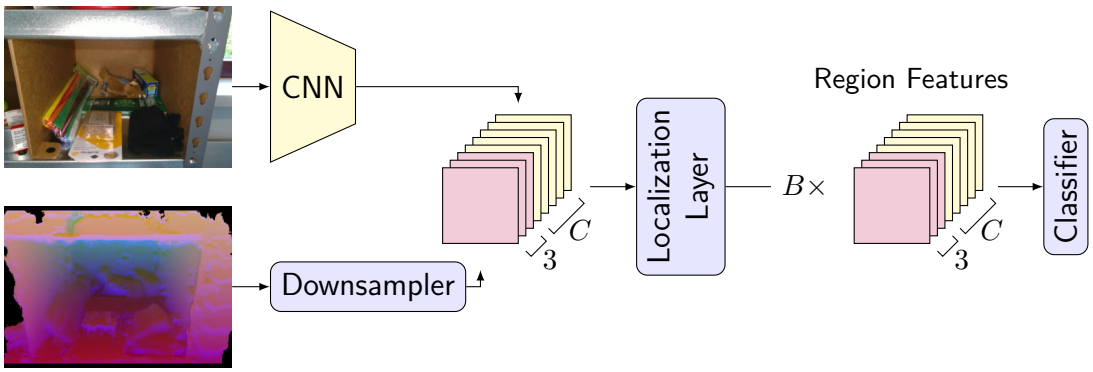


Figure 5.7: Detection pipeline with CNN features from RGB and downsampled HHA-encoded depth. C denotes the number of CNN feature maps after the last convolutional layer (512 for VGG-16). The internal proposal generator produces B proposals (1000).

and color (10). The extracted components are filtered by size (10,000 pixels minimum size for 1920×1080 input), converted to their bounding boxes, and appended to the region proposal network results. Figure 5.6 shows example frames with the generated proposals. Note that more sophisticated state-of-the-art region proposal methods can easily be connected to the pipeline.

5.4.2. Additional Feature Maps

As a minimally invasive modification to the network, depth can be provided to the region proposal network and the classifier stage as an additional feature map. To this end, the depth information is converted into a 3-channel representation called HHA (Gupta, Girshick, et al. 2014), which includes disparity, height above ground, and normal angle to gravity. It has been shown that such more geometrically meaningful depth encodings are better accepted by CNNs than raw depth (Gupta, Girshick, et al. 2014; Gupta, Hoffman, and Malik 2015). To fit the network architecture, the HHA-encoded depth is then bilinearly downsampled

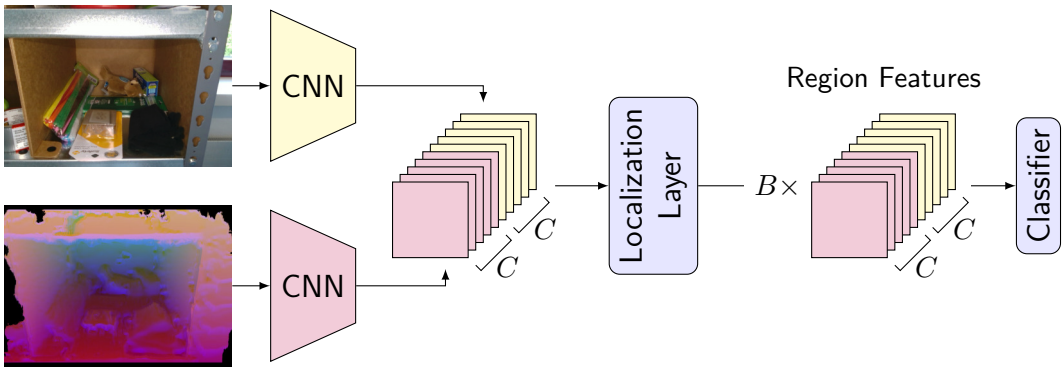


Figure 5.8: Detection pipeline with concatenated CNN features from RGB and HHA-encoded depth. C denotes the number of CNN feature maps after the last convolutional layer (512 for VGG-16). The internal proposal generator produces B proposals (1000).

to the correct feature map size and concatenated to the RGB feature maps (see Figure 5.7).

The first convolutional layer of the region proposal network and the first fully-connected layer of the classification network need to be modified to accept the additional feature maps. Any new weights are initialized from a normal distribution, while preserving the other pretrained weights. Training affects new and old weights as usual.

5.4.3. CNN Features from HHA

Increasing the complexity, one can compute the VGG-16 features not only on the RGB image, but also on the 3-channel HHA image. The resulting feature maps (512 for each modality) can then be concatenated to 1024 feature maps before proceeding with the rest of the DenseCap pipeline (see Figure 5.8).

This approach has the advantage that there should be less information loss than with bilinear downsampling, since the CNN hopefully aggregates information into higher-level features. Indeed, this method has proven useful in multiple contexts (Gupta, Girshick, et al. 2014; Schwarz, Schulz, and Behnke 2015).

5.4.4. Cross Modal Distillation

Using the pretrained VGG-16 network on HHA data is not ideal, since it was not trained for this modality. Training VGG-16 from scratch on depth data is infeasible, since there is no annotated depth dataset which could match ImageNet in the number of classes and annotated frames.

Instead, Gupta, Hoffman, and Malik (2015) propose to use an RGB reference network to generate the supervision data needed for the other modality, a technique they call Cross Modal Distillation. In essence, the RGB network ϕ is computed feed-forward on the RGB frame I_s, generating the target feature maps $\phi(I_s)$. A back-propagation step then trains the depth network ψ on the matching depth frame I_d, minimizing the objective

$$\min_{W_d} \sum_{(I_s, I_d) \in U_{s,d}} ||\psi(I_d) - \phi(I_s)||^2, \tag{5.4}$$

where W_d are the weights of the depth network, and $U_{s,d}$ is the training set of matched RGB and depth frames. Note that no annotation is necessary on $U_{s,d}$, so any RGB-D video (ideally of the target domain) can be used to perform the supervision transfer. In this thesis, the (annotated) training dataset is used for distillation, since more unannotated RGB-D sequences of the target domain are not available.

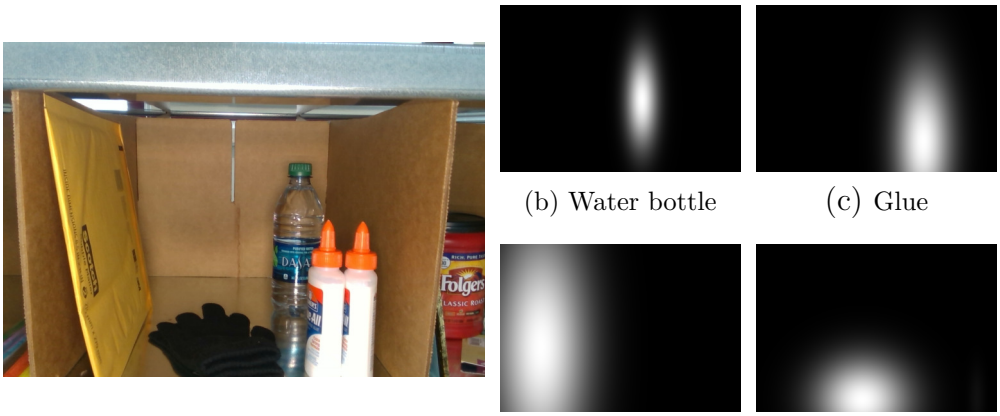
After the initial cross modal distillation training, the trained network can be used in place of the original depth network in Figure 5.8.

5.5. Training Details

As in the original DenseCap work, the ADAM optimizer (Kingma and Ba 2014) is used for training the network with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. However, we adapt a custom learning rate schedule to dampen training oscillations at the end of training (annealing): The learning rate starts at $1 \cdot 10^{-5}$ and is kept constant for 15 epochs, then linearly lowered to $1 \cdot 10^{-6}$ during the next 85 epochs. At 200 epochs, the rate is lowered to $5 \cdot 10^{-7}$, and finally to $1 \cdot 10^{-7}$ at 250 epochs. To prevent overfitting, 50% dropout is used in the entire network.

5.6. Connection with Semantic Segmentation

Semantic segmentation approaches scene perception from another angle - instead of predicting object hypotheses, it strives to classify each pixel of the input image. Our team also developed a semantic segmentation approach for the Amazon Picking Challenge, in order to have a more fine-grained perception suitable for grasp planning. One interesting idea was to improve the semantic segmentation result using the object detection pipeline. This should help to reduce false positives in clutter and help the semantic segmentation settle on the most probable hypothesis.



(a) RGB frame (b) Water bottle (c) Glue (d) Bubble mailer (e) Gloves

Figure 5.9: Probability estimates generated from object detections for an example frame. The object posterior in each pixel is approximated by rendering a Gaussian that corresponds to the bounding box center and extent.

In order to provide guidance to the semantic segmentation, the generated object proposals need to be backprojected to the image to obtain a pixel-wise posterior which can be multiplied with the semantic segmentation posterior before deciding on the most likely class for each pixel.

To this end, Gaussians are rendered for each region proposal, with mean and covariance derived from the box geometry. See Figure 5.9 for an illustration. The Gaussians are summed using the detection scores as weights and the resulting map P_{det} is normalized, i.e. scaled so that the maximum equals one. To allow for detection mistakes, a weak prior is introduced that accounts for false negatives. The final combined posterior is computed as

$$P_{\text{combined}} = P_{\text{seg}}(0.1 + 0.9P_{\text{det}}), \tag{5.5}$$

where P_{seg} is the posterior resulting from semantic segmentation and P_{det} is the estimated posterior from object detection.

While this combination is relatively straightforward (note that the product assumes conditional independence) and the shape approximations by Gaussian masks are rather coarse, this strategy yields a consistent increase in performance nonetheless (see Section 6.1).

6. Evaluation

6.1. APC Dataset

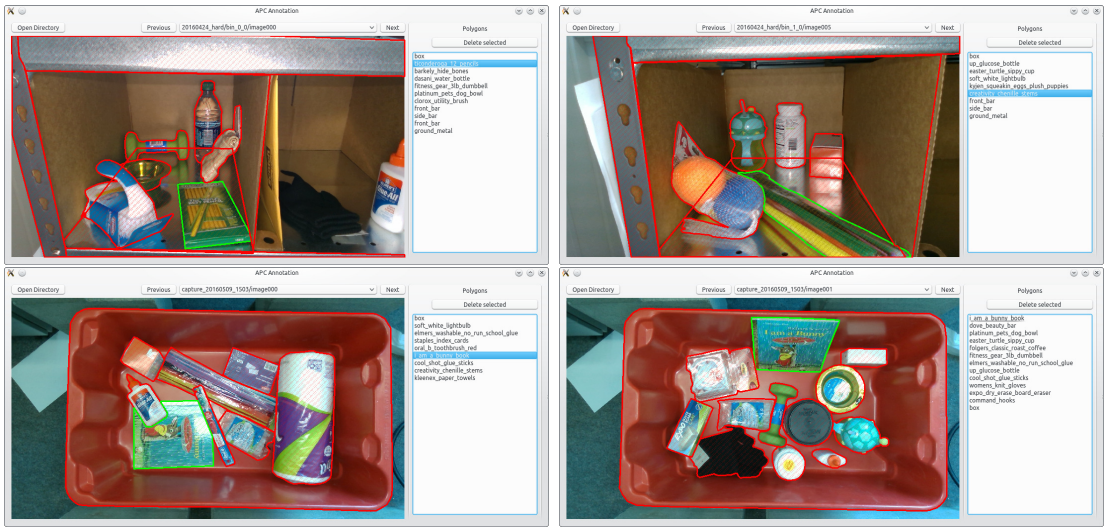


Figure 6.1: Example frames with ground truth annotations from the APC dataset, shown inside the custom annotation tool.

While the APC provided an opportunity to test the entire system under real-world conditions, the object detector can also be evaluated quantitatively on the APC dataset captured by our team. This dataset was also used to train the final models used in the competition. The dataset contains 190 shelf frames, and 117 tote frames, manually annotated using a custom-built annotation tool (see Figure 6.1). The annotations were made on a contour level, since the same dataset is also used for training semantic segmentation, where pixel-wise annotation is required. Other APC teams typically used much larger datasets: Team RBO used 346 annotated frames for their 2015 entry (Jonschkowski et al. 2016), while team Delft annotated 500 frames, and synthesized 20,000 frames of the objects in different orientations for the APC 2016 (Hernandez et al. 2016). The low number of frames highlights the efficiency of the transfer-learning approach, which makes use of the pretraining on ImageNet (VGG-16 CNN) and the Visual Genome Dataset (entire DenseCap pipeline).

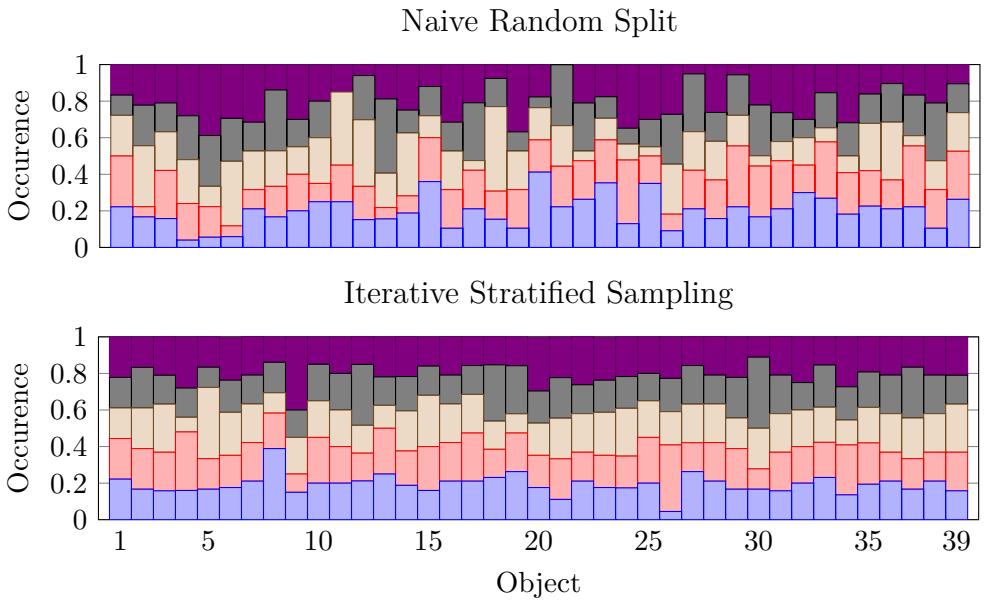


Figure 6.2: Cross validation splits for the APC shelf dataset. The plots show the distribution of labels across the five splits (as five different colors) for each of the 39 objects. The top row shows a naive random split, while the bottom row shows the result of using the Iterative Stratified Sampling algorithm.

For evaluation, we define five-fold cross validation splits on the shelf and tote datasets. Contrary to image classification, splitting the dataset is not straightforward, since each frame can have multiple object labels, but the examples should be distributed as evenly as possible. Following the work of Sechidis, Tsoumakas, and Vlahavas (2011), the Iterative Stratified Sampling algorithm is used to calculate the splits. Figure 6.2 shows a visual comparison of a naive random split and the Iterative Stratified Sampling results. While one can qualitatively observe that the distribution is more even, the algorithm also lowers the LD measure[1] from $9 \cdot 10^{-3}$ to $5 \cdot 10^{-3}$ on the shelf dataset. Note also that a random split may contain splits with no examples for a particular object, which would complicate the evaluation.

To synthetically increase the number of training examples, the images are randomly flipped around the vertical axis during training. Other operations such as scaling, random cropping, and HSV color scaling did not yield a significant improvement.

[1]The LD measure is defined as the mean deviation of the ratio of positive to negative examples for each label over the dataset splits. For details refer to Sechidis, Tsoumakas, and Vlahavas (2011).

Table 6.1: Evaluation of object detection architectures on the shelf dataset. The mAP score is reported for the uninformed and informed case.

		mAP		
Input	Variant	Uninf.	Inf.	F1
RGB	SVM (plain)	–	28.83	68.50
RGB	SVM (tailor)	–	28.87	68.35
RGB	Softmax (no augmentation)	86.04	88.97	76.88
RGB	Softmax (with augmentation)	86.49	89.56	77.10
RGB-D (TGV)	HHA Features (Sec. 5.4.2)	86.53	89.81	77.58
RGB-D (TGV)	Ext. Proposals (Sec. 5.4.1)	87.01	89.84	77.46
RGB-D (TGV)	HHA CNN (Sec. 5.4.3)	86.47	90.12	78.98
RGB-D (TGV)	Distillation (Sec. 5.4.4)	**87.87**	**91.19**	**79.84**
RGB-D (single)[1]	Distillation (Sec. 5.4.4)	86.50	90.13	78.71
RGB-D (DT)[2]	Distillation (Sec. 5.4.4)	87.48	90.32	78.85

[1] Without depth fusion, only from upper camera. Filled using TGV method.
[2] Old filling method used during APC 2016 based on Domain Transform.

6.1.1. Design Choices: Classifier and Depth Inclusion

In order to evaluate the design choices offered in Chapter 5, in particular the classifier and the method of incorporating depth, full cross-validated training and evaluation runs were completed for each choice. Table 6.1 shows the results of this effort. We show the general uninformed mAP score, the informed mAP score (i.e. object classes not present in the scene masked out), and the F1 localization score.

At the first glance, we can tell that the softmax variant with its ability to fine-tune the entire network including the region proposal is far superior to using the fixed network with a trained SVM classifier. In particular, the SVM classifier is bad at ranking the detections across images, which is evident in the mAP metric. A calibration step (e.g. Platt scaling) could improve this behavior. Note that the F1 score, which is more relevant in the APC scenario, is closer to the rest of the methods, but still suboptimal. Training the SVM on-the-fly for just the items in the current shelf ("tailor" variant) makes little difference in both metrics. All remaining tests were performed with the superior softmax classifier. Data augmentation (random image mirroring) slightly improves performance, so all other tests were performed with augmentation.

As expected, incorporating depth results in better performance. The external RGB-D proposal generator (Section 5.4.1) is better than naive HHA concatena-

tion (Section 5.4.2). However, using the CNN for depth feature computation (Section 5.4.3) outperforms the proposal generator. Finally, training the depth CNN using Cross Modal Distillation (Section 5.4.4) gives the best results.

Also, the TGV-regularized method for fusing the two RGB-D streams described in Section 5.3 is superior to the smoothing implementation using Domain Transform, which is again superior to using just one of the two cameras. While capturing correct geometry for grasp and motion planning was certainly a large motivation for using the two-camera setup, it is nonetheless good to see an improvement for object detection as well.

Note that the final improvement using depth relative to the RGB baseline is small (around two percent for all metrics). One explanation for this might be that in this constrained situation, the scene can be very well understood from RGB alone, so that depth adds little additional information. In particular, there are no objects in the object set that differ only by geometry and not by texture. Also, object boundaries can usually be determined using color boundaries, since the brown shelf background is distinctive for most objects.

6.1.2. Final Results

Table 6.2: Final results on the APC dataset.

	mAP		F1
Dataset	Uninformed	Informed	
Shelf	87.87	91.19	79.84
Tote	87.00	88.65	77.90

Final mAP scores can be seen in Table 6.2. As expected, the information which objects are present in the scene greatly improves the mAP results. Table 6.3 illustrates that mostly items which are easily confused (colored toothbrushes, different books from the side) or largely featureless items (glue sticks) benefit from this masking operation.

As detailed in Section 2.2.2, the mAP score itself does not focus on precise localization, so we also give mean F1 score results in Table 6.2. Note that the F1 metric as introduced in Section 2.2.2 is only defined for the informed case. The distribution of the mAP and F1 scores across the objects is shown in Figure 6.3. The detector performance is good for most objects, with a few particularly bad objects. It struggles with very small objects, or objects for which the axis-aligned bounding box is not a good approximation of the shape (e.g. a toothbrush in

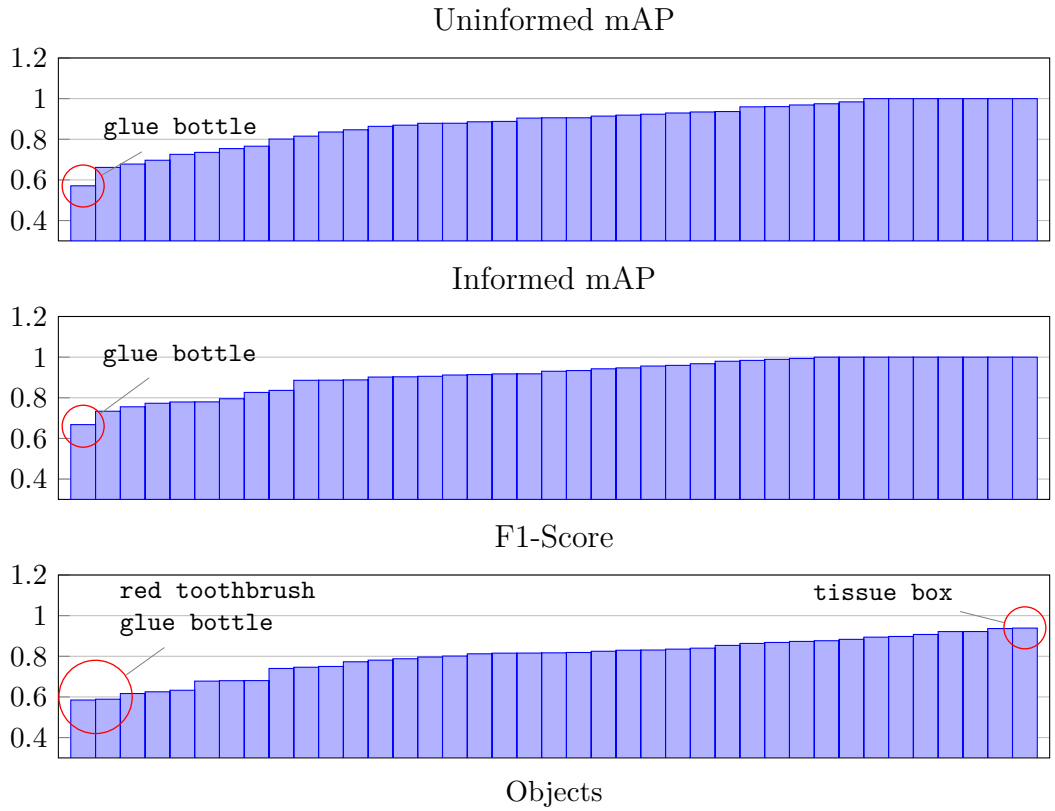


Figure 6.3: Score distribution over the objects. Results are averaged over the cross validation splits. Note that the results for each metric have been sorted independently to show the shape of each distribution (see Appendix A for the source data).

Table 6.3: Effect of the item information on the AP scores. Shown are the top five items which benefit most from masking using the information which items are present in the frame.

	Average Precision			
Object	Uninf.	Inf.	Gain	F1-Score
`i_am_a_bunny_book`	0.84	0.97	+0.13	0.84
`oral_b_toothbrush_green`	0.66	0.78	+0.12	0.68
`cool_shot_glue_sticks`	0.82	0.93	+0.12	0.75
`elmers_washable_no_run_school_glue`	0.57	0.67	+0.10	0.59
`oral_b_toothbrush_red`	0.70	0.77	+0.08	0.58

Table 6.4: Perception runtimes.

			Detector network	
Phase	RGB-D proposal	SVM	RGB	RGB-D
Train	-	3.3 s	45 min	4.5 h[1]
Test	1006 ms	342 ms	340 ms	400 ms

[1] Excluding Cross Modal Distillation (10 min).

some orientations, or the glue bottle). On the other hand, large and cubic-shaped objects (e.g. the tissue box) give the best F1 score.

Runtimes of the various methods are shown in Table 6.4. While training of the RGB-only detector is comparably fast, the RGB-D network is much slower to train due to the larger model size (235,236,725 parameters vs. 131,296,629 parameters). The non-linear increase in computation time is probably stemming from the GPU memory usage, which is very near the limit of 12 GiB for the large model. The training time could be reduced further using precomputation of the VGG CNN features on the training samples, which would also preserve GPU memory. All methods can predict reasonably fast and are suitable for live usage in a robotic system.

Figure 6.4 shows exemplary scenes from the dataset with object detection results.

6.1.3. Combination with Semantic Segmentation

Table 6.5: F1 scores for semantic segmentation.

	Shelf		Tote	
Method	Uninf.	Inf.	Uninf.	Inf.
Segmentation	0.757	0.787	0.789	0.816
Combination[1]	0.782	0.813	0.823	0.833

[1] Finetuned CNN + Segmentation.

The combination described in Section 5.6 leads to a small but consistent increase in performance (see Table 6.5). The object detection results mainly help the segmentation to settle on the correct candidate, thus suppressing false positives.

Figure 6.4: Object detection examples. Left column: Dataset frames with annotated object contours (red). Right column: Output boxes. All shown detections were correctly classified.

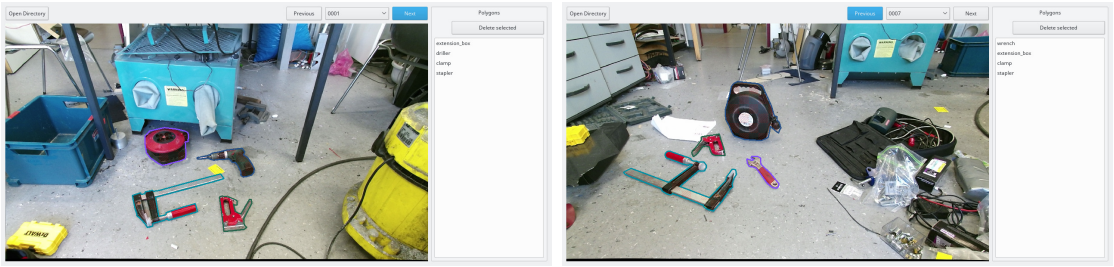


Figure 6.5: Two exemplar frames from the disaster response dataset.

6.2. Disaster Response

In disaster response scenarios, robots can execute missions which would otherwise endanger human response teams. Such dangerous situations could include searching for victims in unstable areas after earthquakes, explosions, floods, or nuclear disasters. The CENTAURO project[2], financed by the European Union, aims to develop a robotic system for disaster response, teleoperated by a human operator using an exoskeleton interface. One idea for easing the load on the operator is that known objects for which interaction plans exist (e.g. a door handle that needs to be turned or a power drill for drilling a hole into a wall) are automatically recognized by the system and presented to the operator as triggerable actions. By automatically performing these otherwise rather tedious tasks, more operator and robot time can be spent on more difficult tasks.

In this context, a dataset has been recorded at University of Bonn for evaluating such ideas. It contains 127 manually annotated RGB-D frames captured using a Kinect version 2 camera inside a cluttered mechanics workshop. Six object classes are annotated using the same methodology introduced in Section 6.1: Five mechanical tools (clamp, driller, extension box, stapler, wrench) and door handles. Figure 6.5 shows exemplary frames from the dataset.

Table 6.6 details the detection results on the dataset. Again, Iterative Stratified Sampling has been used to create five splits. The baseline RGB performance is worse than on the APC dataset. While the dataset may be harder since objects are often very small (4% of the image width for certain door knobs) and the scenes are heavily cluttered, investigation revealed that the fixed anchor box shapes in the DenseCap region proposal network do not represent the distribution of shapes present in the dataset. To rectify this, we follow an idea of Redmon et al. (2015) and use K-Means clustering on the ground truth box shapes (width and height) to obtain better anchor box shapes. This gives a roughly 6% increase in mAP (see Table 6.6). As in the APC case, incorporating depth increases performance.

[2] `http://www.centauro-project.eu`

Table 6.6: Object detection results on the disaster response dataset.

Input	Variant	mAP	F1
RGB	-	68.79	63.34
RGB	Box shape clustering[1]	74.82	68.31
RGB-D (TGV)	Distillation (Section 5.4.4)	72.28	65.90
RGB-D (TGV)	Dist. + clustering	**78.88**	**69.34**

[1] Replace fixed anchor box shapes with shapes extracted from dataset (see text).
[2] Depth measurements filled using the guided upsampling method described in Section 5.3.

Table 6.7: AP and F1 scores per object on the disaster response dataset.

Measure	Clamp	Door handle	Driller	Extension box	Stapler	Wrench
AP	0.823	0.541	0.873	1.000	0.848	0.649
F1	0.712	0.399	0.763	0.923	0.736	0.627

Combining both improvements yields the final mAP of 78.88. Figure 6.6 shows exemplary detector output.

Table 6.7 shows the score distribution across the object classes. It can be seen that while all other classes obtain good results, the wrench and in particular the door handle classes perform worse. This may be due to the number of examples: Two different door handle types were trained (knob and handle) with only twelve examples in total. In comparison, the stapler class has 66 examples. For future work with this dataset, more training examples should be recorded, especially for such classes with diverse instances.

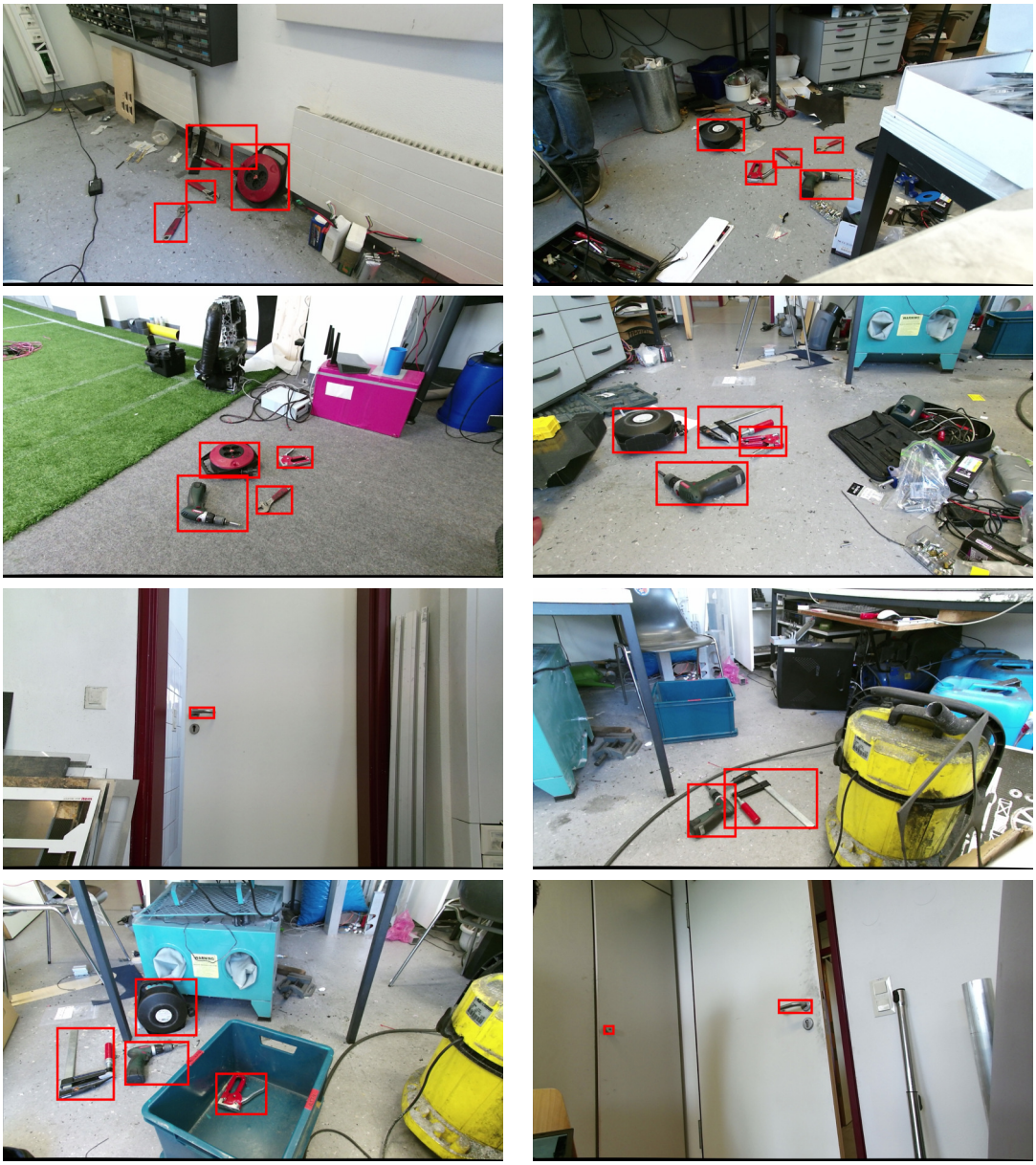


Figure 6.6: Object detection examples on disaster response dataset

7. System Integration for the APC 2016

7.1. Motion Generation

As already mentioned, the kinematic constraints imposed by the APC shelf are severe. The available space is very narrow, and objects can be partially occluded, meaning that the robot has to reach around other objects. We will now show how feasible grasps can be heuristically selected, how collision-free robot configurations can be found, and how motions can be generated without complex motion planning.

7.1.1. Heuristic Grasp Selection

Some objects (in particular big objects) require grasping at specific points to ensure that the grasp is successful and the object can be lifted in a controlled manner. For such objects, an ICP-based registration with a prerecorded model of the object is performed, which is not the subject of this thesis. For details, see Schwarz, Milan, et al. (2017). Note that only three objects required this special procedure: The duct tape, the pack of tube socks, and the paper towel roll.

For all other objects, grasp poses are selected heuristically. The system can generate two basic grasps: A top grasp and a center grasp.

The top grasp tries to grasp the 3D bounding box of the object from above. Using the semantic segmentation result, the object point, whose ground projection is closest to the projection of the 3D bounding box center is selected. The grasp height is chosen as the maximum height of object points in a cylinder around the chosen position. The grasp position is then refined to the next object point. The grasp is always performed from above the object, pointing straight down.

Center grasps are found using the 2D image-space bounding box. Again, the object point closest to the box center is chosen, this time in image space. The grasp direction is determined by the surface normal estimated in the local neighborhood.

Figure 7.1 shows top and center grasps on exemplary scenes.

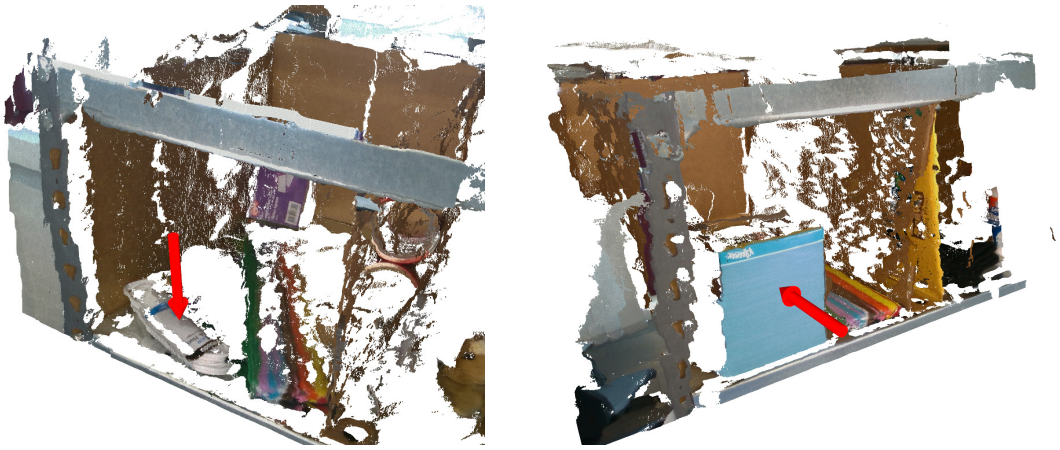


Figure 7.1: Heuristic grasp selection. Left: Top grasp on an extension cord. Right: Front grasp on the tissue box.

7.1.2. Inverse Kinematics

Instead of using a generic kinematics solver for motion planning, we can use the geometric constraints given by the arrangement to simplify the planning problem. This idea results in an intelligent kinematics solver driven by two ideas: First, the suction pose itself is invariant to rotations around the suction axis, and second, the solver should resolve the inherent redundancy in the kinematic chain so as to minimize the chance of collisions with the environment.

As a basis, we use a selectively damped least squares (SDLS) solver (Buss and Kim 2005), since it performs well near singularities and finds approximative solutions for non-reachable goals, where other algorithms quickly begin to diverge. We augment it with a null-space optimization step, which projects the gradient of a secondary objective f to the null space of the SDLS Jacobian matrix J. This allows to optimize secondary objectives using the kinematic chain redundancies while still reaching the target pose with the endeffector.

We first define a joint-level null space objective g:

$$\begin{aligned} g_i(q) = {} & w_l \max\{0, q - (q_i^+ - q_\delta)\}^2 \\ & + w_l \min\{0, q - (q_i^- + q_\delta)\}^2 \\ & + w_c (q - q_i^{(c)})^2, \end{aligned} \tag{7.1}$$

where i is the joint index, q is the joint position, q_i^+ and q_i^- are the upper and lower joint limits, q_δ is a joint limit threshold, $q_i^{(c)}$ is the "convenient" configuration for this joint, and w is used to form a linear combination of the costs. As can be seen, this objective prefers a convenient configuration and avoids joint limits.

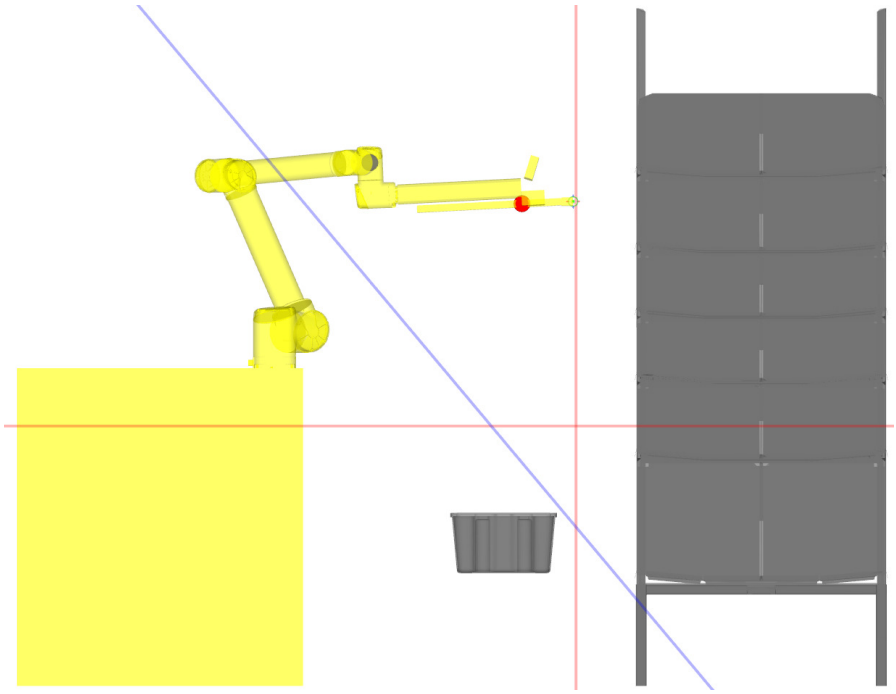


Figure 7.2: Penalizing planes in IK solver. The red/blue planes penalize violation by the red/blue spheres, respectively.

More interestingly in this application, we also specify Cartesian-space costs using a plane-violation model:

$$h_{\vec{n},d}(\vec{x}) = (\max\{0, (-\vec{n}\vec{x}^T + d)\})^2, \tag{7.2}$$

where $\vec{n}$ and d specify an oriented plane $\vec{n}\vec{x}^T - d = 0$, and $\vec{x}$ is some Cartesian point. This model is used to avoid specified half-spaces with parts of the robot (e.g. do not enter the shelf with the cameras).

Finally, we obtain the combined costs f:

$$f(\vec{q}, \vec{x_l}, \vec{x_w}) = \sum_{i \in Q} g_i(\vec{q_i}) + h_{\vec{n}_s,d_s}(\vec{x_l}) + h_{\vec{n}_t,d_t}(\vec{x_l}) + h_{\vec{n}_b,d_b}(\vec{x_w}), \tag{7.3}$$

where $\vec{q}$ is the vector of joint positions, $\vec{x}_l$ and $\vec{x}_w$ are Cartesian positions of the linear extension and the camera module, and $\vec{n_i}, d_i$ describe three half spaces which are avoided (see Figure 7.2). This half space penalization ensures that we do not enter the shelf with the cameras, that the linear extension is horizontal during manipulation in the shelf[1], and that collisions with the robot base are avoided.

[1]This also uses the penalization of linear extension in Equation (7.1).

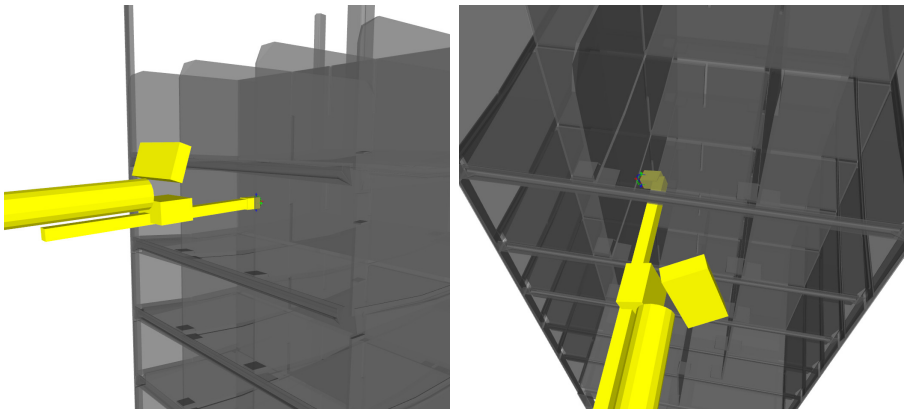


Figure 7.3: Nullspace-optimizing IK. Left: Front grasp. Right: Side grasp.

One iteration of the solver calculates the update δ_q as follows:

$$\bar{J}, \bar{J}^{+} = \text{SDLS}(RPR^{T}J) \tag{7.4}$$

$$N = I - \bar{J}^{+}\bar{J} \tag{7.5}$$

$$\delta_q = \bar{J}^{+}\Delta_x - \alpha N \nabla f(\vec{q}, \vec{x_l}, \vec{x_w}), \tag{7.6}$$

where R is the target orientation of the endeffector, P is a projector zeroing the roll component (allowing rotation around the suction axis), J is the $6 \times n$ kinematic Jacobian matrix, N is the null space projector of $\bar{J}$, Δ_x is the remaining 6D pose difference, and α is the step size for null space optimization.

Using this custom IK solver, it is possible to reach difficult target poses in the shelf and tote without collisions (see Figure 7.3).

7.1.3. Parametrized Motion Primitives

In earlier works (Schwarz, Rodehutskors, et al. 2016), we developed a keyframe-based interpolation system. Keyframes specify joint- or Cartesian space configurations of the robot (or sub-groups like the endeffector). Each keyframe also contains joint and Cartesian velocity and acceleration limits, which constrain the motion to the keyframe pose. Motions can be edited in a dedicated 3D GUI, which is particularly useful for fixed motions, such as approaching shelf bins for perception or dropping items into the tote. Other more dynamic motions, like object picking motions, are created on the fly. Based on the selected grasp pose, Cartesian keyframes are placed in pregrasp, grasp and retract poses. Finally, the system smoothly interpolates between the keyframe configurations in joint or Cartesian space to generate trajectories. The trajectory is then executed on the robot. Since collisions with objects may still be possible, the UR10 arm is configured to stop on

Table 7.1: Final scores of the top four teams for each task.

Stowing		Picking	
Team	Score	Team	Score
Delft (Hernandez et al. 2016)	214	Delft (Hernandez et al. 2016)	105
NimbRo Picking (ours)	186	PFN	105
MIT	164	Nimbro Picking (ours)	97
PFN	161	MIT	67

unexpected contact forces. The software can then cancel the trajectory, re-enable the arm and execute a retract trajectory. Failed objects are retried at the end of the picking sequence.

7.2. Overall System Performance

During the APC 2016, the object detection pipeline was in an earlier state than the best architectures described in Chapter 5. In particular, depth filling was done using the slightly inferior Domain Transform implementation and the external proposal generator was used to incorporate depth measurements. Finally, the SVM classifier was used for classification.

The system attempted both the picking and the stowing tasks successfully. Table 7.1 shows the final official scores of the top four teams of APC 2016. Videos from both runs as well as an overview video are available online.[2] During the stowing task, the system stowed eleven out of twelve items into the shelf. Sadly, one of the successfully stowed items was misrecognized. Since the system masked out objects already stowed (the informed case), the remaining item (a toothbrush) could not be detected. We anticipated this problem and had built in a fallback mechanism, which reset the mask to all objects after some number of failed attempts, but this failed due to an object size threshold. The misrecognition led to the second place, since team Delft successfully stowed all items.

In our picking run, the system picked ten out of twelve items (see Table 7.2). Due to the difficult arrangements of items in the shelf bins, the robot dropped three items during retraction from the bin, since they collided with other items. However, the items fell into the tote and were thus counted as successful picks. Unfortunately, the system judged that the drop happened *inside* the shelf bin and marked the items as present in the shelf in the output file. The resulting penalties

[2]Overview: `https://youtu.be/7Dlt8T3s3HY`, stowing: `https://youtu.be/B6ny9ONfdx4`, picking: `https://youtu.be/q9YiD8OvwDc`.

Table 7.2: Picking Run at APC 2016

Bin	Item	Pick	Drop	Report
A	duct tape	×	×	×
B	bunny book	✓	✓	×[2]
C	squeaky eggs	✓	×	✓
D	crayons[1]	✓	×	✓
E	coffee	✓	✓	×[2]
F	hooks	✓	×	✓
G	scissors	×	×	×
H	plush bear	✓	×	✓
I	curtain	✓	×	✓
J	tissue box	✓	×	✓
K	sippy cup	✓	×	✓
L	pencil cup	✓	✓	×[2]
	Sum	10	3	7

[1] Misrecognized, corrected on second attempt.
[2] Incorrect report, resulting in penalty.

dropped our score from 152 points to 97 points—just behind the first and second place with both 105 points.

On the final day of the competition, we showed the capabilities of the system in an open demonstration. In a very successful demonstrated picking run, we showcased our ability to handle both "hard" objects with three bonus points: The pencil cup (requires knocking over to be able to suction it) and the heavy dumbbell (requires very precise suction point localization).

8. Conclusion

This thesis presents an end-to-end pipeline for RGB-D preprocessing, object detection, grasp selection and motion execution. Besides the demonstrated integration effort, scientific contributions were the method for fusing two RGB-D streams using guided upsampling, adaption of the DenseCap method for object detection, incorporation of depth measurements, and a custom inverse kinematics solver for reaching into the shelf.

The individual design decisions were evaluated on the APC dataset, and the object detection pipeline shows good performance on both the APC and the disaster response dataset. Finally, the entire system was evaluated at the APC 2016, reaching a very good second and third place. A general description of the entire system has been submitted and accepted for the International Conference on Robotics and Automation (ICRA) 2017 (Schwarz, Milan, et al. 2017).

A possible extension point is the efficient training for new objects, which may be required daily in high-volume warehouses. Of particular interest in this context is keeping the annotation workload low, since this is easily the most time-consuming part of training. In particular, the current system requires manual arrangement of "natural" training scenes with multiple objects.

The Amazon Robotics Challenge (ARC) 2017[1] is the successor of the Amazon Picking Challenge. It raises the difficulty of existing tasks (packing items into cardboard boxes instead of totes), and introduces a new hybrid task (first stowing objects and then picking them into boxes). The shelf is replaced by a participant-designed storage system. For object perception, the main challenge is efficient training: Half of the items to be picked or stowed will be new items, which are provided to the team 30 min before the run starts. Efficiently using this time to capture data, annotate it, and train classifiers will be key to win the competition. Here, a hybrid Softmax/SVM system might be applicable: For the training set available before the run, the pipeline can be trained end-to-end. The pretrained network can then be used to extract feature vectors for fast SVM training on the new objects.

[1] `https://www.amazonrobotics.com/#/pickingchallenge`

A. Detailed APC results

Table A.1: Distribution of Average Precision and F1-Score over the objects of the APC shelf dataset.

	Average Precision			
Object	Uninf.	Inf.	Gain	F1-Score
barkely_hide_bones	0.906	0.959	+0.054	0.816
cherokee_easy_tee_shirt	0.736	0.756	+0.020	0.625
clorox_utility_brush	0.888	0.903	+0.015	0.801
cloud_b_plush_bear	0.975	0.980	+0.005	0.835
command_hooks	0.678	0.734	+0.056	0.617
cool_shot_glue_sticks	0.815	0.931	+0.115	0.750
crayola_24_ct	0.936	0.956	+0.020	0.854
creativity_chenille_stems	0.801	0.826	+0.025	0.680
dasani_water_bottle	0.879	0.887	+0.008	0.815
dove_beauty_bar	0.934	1.000	+0.066	0.877
dr_browns_bottle_brush	0.879	0.888	+0.009	0.788
easter_turtle_sippy_cup	1.000	1.000	+0.000	0.898
elmers_washable_no_run_school_glue	0.571	0.668	+0.097	0.589
expo_dry_erase_board_eraser	0.919	0.943	+0.024	0.831
fiskars_scissors_red	0.726	0.779	+0.054	0.633
fitness_gear_3lb_dumbbell	1.000	1.000	+0.000	0.873
folgers_classic_roast_coffee	0.969	0.989	+0.020	0.937
hanes_tube_socks	0.869	0.902	+0.033	0.817
i_am_a_bunny_book	0.836	0.968	+0.132	0.840
jane_eyre_dvd	0.847	0.918	+0.071	0.773
kleenex_paper_towels	1.000	1.000	+0.000	0.939
kleenex_tissue_box	1.000	1.000	+0.000	0.922
kyjen_squeakin_eggs_plush_puppies	0.914	0.914	+0.000	0.796
laugh_out_loud_joke_book	0.923	0.934	+0.011	0.830
oral_b_toothbrush_green	0.662	0.780	+0.118	0.678
oral_b_toothbrush_red	0.697	0.773	+0.076	0.585
peva_shower_curtain_liner	0.929	0.947	+0.018	0.863
platinum_pets_dog_bowl	1.000	1.000	+0.000	0.883
rawlings_baseball	0.886	0.886	+0.000	0.781
rolodex_jumbo_pencil_cup	1.000	1.000	+0.000	0.907
safety_first_outlet_plugs	0.961	0.994	+0.033	0.868
scotch_bubble_mailer	0.864	0.912	+0.048	0.819
scotch_duct_tape	0.906	0.906	+0.000	0.825
soft_white_lightbulb	0.960	1.000	+0.040	0.894
staples_index_cards	0.754	0.795	+0.041	0.746
ticonderoga_12_pencils	0.904	0.918	+0.014	0.740
up_glucose_bottle	1.000	1.000	+0.000	0.922
womens_knit_gloves	0.984	0.984	+0.000	0.812
woods_extension_cord	0.766	0.836	+0.071	0.680

Table A.2: Distribution of Average Precision and F1-Score over the objects of the APC tote dataset.

	Average Precision			
Object	Uninf.	Inf.	Gain	F1-Score
barkely_hide_bones	0.891	0.891	+0.000	0.817
cherokee_easy_tee_shirt	0.835	0.892	+0.057	0.741
clorox_utility_brush	0.974	0.977	+0.003	0.891
cloud_b_plush_bear	0.934	0.934	+0.000	0.803
command_hooks	0.954	0.969	+0.015	0.827
cool_shot_glue_sticks	0.790	0.845	+0.055	0.726
crayola_24_ct	0.900	0.908	+0.008	0.790
creativity_chenille_stems	0.718	0.737	+0.019	0.644
dasani_water_bottle	0.954	0.955	+0.001	0.864
dove_beauty_bar	0.960	0.973	+0.013	0.884
dr_browns_bottle_brush	0.946	0.953	+0.007	0.833
easter_turtle_sippy_cup	1.000	1.000	+0.000	0.875
elmers_washable_no_run_school_glue	0.779	0.792	+0.013	0.682
expo_dry_erase_board_eraser	0.846	0.859	+0.014	0.757
fiskars_scissors_red	0.469	0.514	+0.045	0.357
fitness_gear_3lb_dumbbell	0.906	0.948	+0.042	0.770
folgers_classic_roast_coffee	1.000	1.000	+0.000	0.930
hanes_tube_socks	0.690	0.691	+0.001	0.756
i_am_a_bunny_book	0.914	0.929	+0.015	0.790
jane_eyre_dvd	1.000	1.000	+0.000	0.900
kleenex_paper_towels	0.941	0.967	+0.026	0.808
kleenex_tissue_box	0.962	0.963	+0.001	0.878
kyjen_squeakin_eggs_plush_puppies	0.934	0.934	+0.000	0.836
laugh_out_loud_joke_book	0.934	0.934	+0.000	0.832
oral_b_toothbrush_green	0.734	0.753	+0.020	0.652
oral_b_toothbrush_red	0.340	0.386	+0.047	0.302
peva_shower_curtain_liner	0.737	0.750	+0.013	0.643
platinum_pets_dog_bowl	0.974	1.000	+0.026	0.893
rawlings_baseball	0.937	0.951	+0.014	0.818
rolodex_jumbo_pencil_cup	1.000	1.000	+0.000	0.901
safety_first_outlet_plugs	0.680	0.690	+0.010	0.557
scotch_bubble_mailer	0.864	0.869	+0.005	0.788
scotch_duct_tape	0.952	0.967	+0.015	0.825
soft_white_lightbulb	0.898	0.947	+0.049	0.799
staples_index_cards	0.801	0.822	+0.022	0.751
ticonderoga_12_pencils	0.916	0.934	+0.018	0.882
up_glucose_bottle	0.864	0.938	+0.074	0.806
womens_knit_gloves	1.000	1.000	+0.000	0.890
woods_extension_cord	1.000	1.000	+0.000	0.880

List of Figures

List of Tables

Bibliography

Berner, Alexander, Jun Li, Dirk Holz, Jörg Stückler, Sven Behnke, and Reinhard Klein (2013). "Combining contour and shape primitives for object detection and pose estimation of prefabricated parts". In: *Proceedings of the IEEE international conference on image processing*, pp. 3326–3330.

Buchholz, D., D. Kubus, I. Weidauer, A. Scholz, and F. M. Wahl (2014). "Combining visual and inertial features for efficient grasping and bin-picking". In: *Proceedings of the IEEE international conference on robotics and automation*, pp. 875–882.

Buss, Samuel R and Jin-Su Kim (2005). "Selectively damped least squares for inverse kinematics". In: *Graphics, gpu, and game tools* 10.3, pp. 37–49.

Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille (2015). "Semantic image segmentation with deep convolutional nets and fully connected CRFs". In: *International conference on learning representations (ICLR)*.

Correll, Nikolaus, Kostas E Bekris, Dmitry Berenson, Oliver Brock, Albert Causo, Kris Hauser, Kei Okada, Alberto Rodriguez, Joseph M Romano, and Peter R Wurman (2016). "Lessons from the Amazon Picking Challenge". In: *Arxiv preprint arxiv:1601.05484*.

Domae, Y., H. Okuda, Y. Taguchi, K. Sumi, and T. Hirai (2014). "Fast graspability evaluation on single depth maps for bin picking with general grippers". In: *Proceedings of the IEEE international conference on robotics and automation*, pp. 1997–2004.

Drost, B., M. Ulrich, N. Navab, and S. Ilic (2010). "Model globally, match locally: efficient and robust 3D object recognition". In: *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*, pp. 998–1005.

Eppner, Clemens, Sebastian Höfer, Rico Jonschkowski, Roberto Martín-Martín, Arne Sieverling, Vincent Wall, and Oliver Brock (2016). "Lessons from the Amazon Picking Challenge: Four aspects of building robotic systems". In: *Proceedings of robotics: science and systems*. AnnArbor, Michigan.

Everingham, Mark, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman (2010). "The pascal visual object classes (VOC) challenge". In: *International journal of computer vision* 88.2, pp. 303–338.

Ferstl, David, Christian Reinbacher, Rene Ranftl, Matthias Rüther, and Horst Bischof (2013). "Image guided depth upsampling using anisotropic total generalized variation". In: *Proceedings of the fourteenth IEEE international conference on computer vision*, pp. 993–1000.

Gastal, Eduardo SL and Manuel M Oliveira (2011). "Domain transform for edge-aware image and video processing". In: *ACM transactions on graphics (ToG)*. Vol. 30. 4. ACM, p. 69.

Geiger, Andreas, Martin Roser, and Raquel Urtasun (2010). "Efficient large-scale stereo matching". In: *Proceedings of the tenth asian conference on computer vision*.

Girshick, Ross (2015). "Fast R-CNN". In: *Proceedings of the fifteenth IEEE international conference on computer vision*. Santiago de Chile, Chile.

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

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey E. Hinton (2013). "Speech recognition with deep recurrent neural networks". In: *IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 6645–6649.

Gupta, Saurabh, Ross Girshick, Pablo Arbeláez, and Jitendra Malik (2014). "Learning rich features from rgb-d images for object detection and segmentation". In: *European conference on computer vision (ECCV)*. Springer, pp. 345–360.

Gupta, Saurabh, Judy Hoffman, and Jitendra Malik (2015). "Cross modal distillation for supervision transfer". In: *Arxiv preprint arxiv:1507.00448*.

Hernandez, Carlos, Mukunda Bharatheesha, Wilson Ko, Hans Gaiser, Jethro Tan, Kanter van Deurzen, Maarten de Vries, Bas Van Mil, Jeff van Egmond, Ruben Burger, et al. (2016). "Team delft's robot winner of the amazon picking challenge 2016". In: *Arxiv preprint arxiv:1610.05514*.

Hoffman, Judy, Saurabh Gupta, Jian Leong, Sergio Guadarrama, and Trevor Darrell (2016). "Cross-modal adaptation for RGB-D detection". In: *Proceedings of the IEEE international conference on robotics and automation*. IEEE, pp. 5032–5039.

Johnson, Justin, Andrej Karpathy, and Li Fei-Fei (2016). "DenseCap: Fully convolutional localization networks for dense captioning". In: *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*.

Jonschkowski, Rico, Clemens Eppner, Sebastian Höfer, Roberto Martín-Martín, and Oliver Brock (2016). "Probabilistic multi-class segmentation for the amazon picking challenge". In: *Intelligent robots and systems (IROS), 2016 IEEE/RSJ international conference on*. IEEE, pp. 1–7.

Kaipa, Krishnanand N, Akshaya S Kankanhalli-Nagendra, Nithyananda B Kumbla, Shaurya Shriyam, Srudeep Somnaath Thevendria-Karthic, Jeremy A Marvel, and Satyandra K Gupta (2016). "Addressing perception uncertainty induced failure modes in robotic bin-picking". In: *Robotics and computer-integrated manufacturing* 42, pp. 17–38.

Kingma, Diederik and Jimmy Ba (2014). "Adam: a method for stochastic optimization". In: *Arxiv preprint arxiv:1412.6980*.

Krishna, Ranjay, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Fei-Fei Li (2016). "Visual genome: connecting language and vision using crowdsourced dense image annotations". In: *Arxiv preprint arxiv: 1602.07332*.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2012). "ImageNet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. Ed. by P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, pp. 1097–1105.

Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg (2016). "SSD: Single shot multibox detector". In: *Proceedings of the 14th european conference on computer vision*. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-46448-0_2, pp. 21–37. ISBN: 978-3-319-46447-3 978-3-319-46448-0.

Long, Jonathan, Evan Shelhamer, and Trevor Darrell (2015). "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*. Boston, Massachusetts.

Nieuwenhuisen, Matthias, David Droeschel, Dirk Holz, Jörg Stückler, Alexander Berner, Jun Li, Reinhard Klein, and Sven Behnke (2013). "Mobile bin picking with an anthropomorphic service robot". In: *Proceedings of the IEEE international conference on robotics and automation*, pp. 2327–2334.

Pretto, Alberto, Stefano Tonello, and Emanuele Menegatti (2013). "Flexible 3D localization of planar objects for industrial bin-picking with monocamera vision system". In: *IEEE international conference on automation science and engineering (CASE)*, pp. 168–175.

Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi (2015). "You only look once: unified, real-time object detection". In: *Arxiv preprint arxiv: 1506.02640*.

Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun (2015). "Faster R-CNN: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems 28: annual conference on neural information processing systems*, pp. 91–99.

Russakovsky, Olga, 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 (2015). "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* 115.3, pp. 211–252.

Schwarz, Max, Anton Milan, Christian Lenz, Aura Muñoz, Arul Selvam Periyasamy, Michael Schreiber, Sebastian Schüller, and Sven Behnke (2017). "NimbRo picking: versatile part handling for warehouse automation". In: *Accepted for 2017 IEEE international conference on robotics and automation (ICRA)*. IEEE.

Schwarz, Max, Tobias Rodehutskors, David Droeschel, Marius Beul, Michael Schreiber, Nikita Araslanov, Ivan Ivanov, Christian Lenz, Jan Razlaw, Sebastian

Schüller, David Schwarz, Angeliki Topalidou-Kyniazopoulou, and Sven Behnke (2016). "NimbRo Rescue: solving disaster-response tasks through mobile manipulation robot Momaro". In: *Accepted for journal of field robotics (JFR), available at* `http://www.ais.uni-bonn.de/papers/JFR_NimbRo_Rescue_Momaro.pdf`.

Schwarz, Max, Hannes Schulz, and Sven Behnke (2015). "RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features". In: *Proceedings of the IEEE international conference on robotics and automation*. IEEE, pp. 1329–1335.

Sechidis, Konstantinos, Grigorios Tsoumakas, and Ioannis Vlahavas (2011). "On the stratification of multi-label data". In: *Joint european conference on machine learning and knowledge discovery in databases*. Springer, pp. 145–158.

Simonyan, K. and A. Zisserman (2014). "Very deep convolutional networks for large-scale image recognition". In: *Corr* abs/1409.1556.

Song, S., S. P. Lichtenberg, and J. Xiao (2015). "SUN RGB-D: A RGB-D scene understanding benchmark suite". In: *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*, pp. 567–576.

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*, pp. 3104–3112.

Viola, Paul and Michael Jones (2001). "Rapid object detection using a boosted cascade of simple features". In: *Computer vision and pattern recognition, 2001. cvpr 2001. proceedings of the 2001 ieee computer society conference on*. Vol. 1. IEEE, pp. I–I.

Yu, Kuan-Ting, Nima Fazeli, Nikhil Chavan-Dafle, Orion Taylor, Elliott Donlon, Guillermo Diaz Lankenau, and Alberto Rodriguez (2016). "A summary of team MIT's approach to the Amazon Picking Challenge 2015". In: *Arxiv preprint arxiv:1604.03639*.