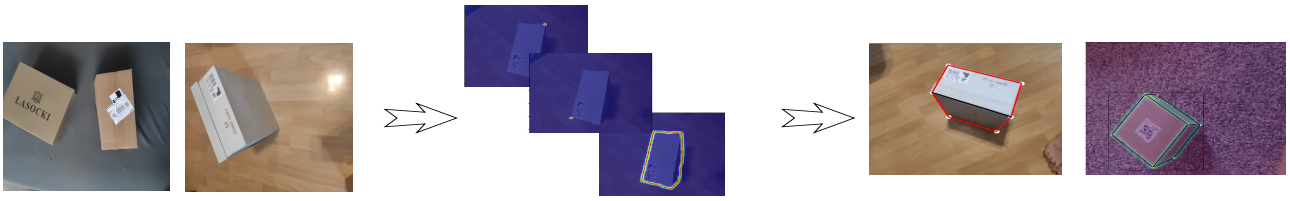
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2021
http://excel.fit.vutbr.cz

Detection of Boxes in Image

Adam Žitňanský*



Abstract
This work addresses the problem of *cuboid detection*, more specifically detection of *boxes* in RGB images. The main result is the implementation of a system for detecting boxes and their accurate localization based on the detection of cuboid primitives: corners and edge points.
The system consists of a detector of corner and edge points based on convolutional neural networks. Such detected primitives are processed into a model of the cuboid. Our system was trained and evaluated on a custom dataset of packaging boxes, which was created as a part of this work. The trained model achieved PCK 90 % and recall 86 % for corners resp. 97.5 % and 96 % for edge points on unseen data.

Keywords: Shape Model Detection — Cuboid Detection — Stacked Hourglass Network

Supplementary Material: Github repository

*xzitna02@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Every day billions of euros worth of goods are packed and transported all over the world. A lot of them are packed in boxes. This makes boxes detection an actual topic, which could be helpful in the process of automation in goods transportation, packaging and many other sectors where boxes are heavily used.

The aim of this paper is to provide a proposal and implementation of a system for detecting boxes based on key point detection. Unlike bounding box detectors, which directly detect the rectangular area covering the object, key point detectors allows to more precisely describe the position of the box by detecting its vertices and edges. This could be useful in applications where orientation and dimensions of the box matters.

2. Background

This work is specialised on cardboard boxes, making it difficult to directly compare it to existing systems. It can be assumed that there are some proprietary systems designed for this specific task, but when it comes to open source and published papers, only more general solutions are available.

The available systems suitable for this task could be divided into two categories. The first approach is using general purpose detectors such as RCNN [1], YOLO [2] and SSD [3] family bounding-box regression detectors fine-tuned for detection of boxes. As stated earlier, the main disadvantage of this approach is that the axis-aligned bounding box representation is missing information about orientation of the box and dimensional information is also limited.

An alternative approach is using a general purpose cuboid detector like the one presented by Dwibed et al. [4]. This detector uses a convolutional neural network for direct regression of cuboid corner coordinates, and thus it fully describes the position of a cuboid similarly to what we want to achieve. The main problem is the absence of source code and a difference in the

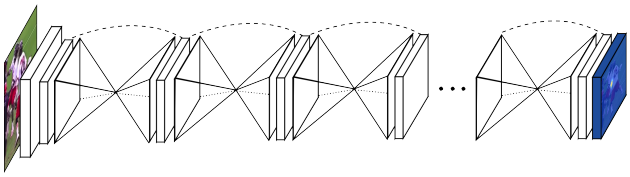


Figure 1. Scheme of the Stacked Hourglass Network. The network consists of multiple hourglass modules stacked one after another. Image taken from [6].

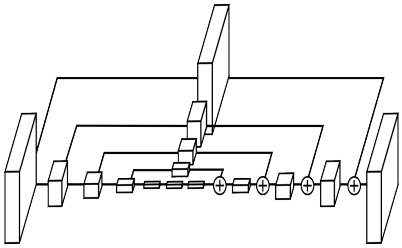


Figure 2. Single Hourglass module, each box represents single residual module (see Fig. 3). Image taken from [6].

purpose (and thus in the dataset). Because of this fact we do not know the performance for cardboard boxes specifically to make it directly comparable with our system.

Our system is using heatmap regression for keypoint localisation which is a widely used method for human pose estimation. In comparison with direct keypoint regression, this approach allows for multiple instances of the key point to be detected. This makes it suitable for bottom-up approach to detection which my system is using. Another reason for this choice was that in the field of human pose estimation [5], heatmap regression has been shown to yield better results than direct keypoint regression and most state of the art solutions are using it. Similarly to how human pose detection systems detect different joints, which are then connected into a model of human torso, our system detects multiple types of corner and edge points and use them to obtain the final output – the shape model of the box.

3. Stacked Hourglass Network

Stacked Hourglass Network [6] is an ConvNet architecture for regression of key point heatmaps, which was originally designed for human pose estimation. As shown in Fig. 1, the network consist of multiple so called hourglass modules, which are stacked one after another. Each hourglass module (shown in Fig. 2) acts as an encoder-decoder block with use of skip connections to preserve spatial information across different resolutions. Stacking multiple modules together, so that the previous module output is the next module's input, with the loss computed for each module's output

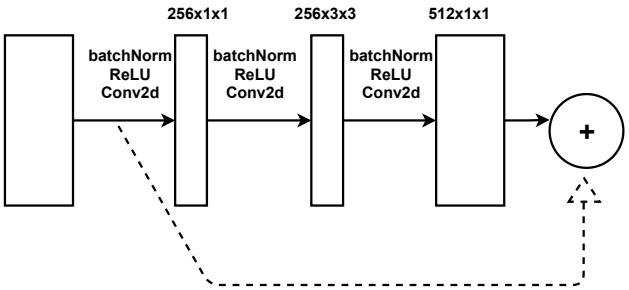


Figure 3. Residual module architecture. Numbers above boxes represents parameters of the convolutional layer (filters × kernel size × strides).

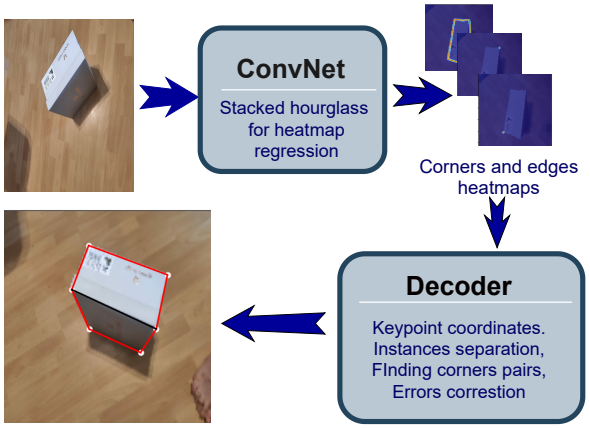


Figure 4. Simplified scheme of the detection pipeline. Blue boxes represent main parts of the system pipeline. An example of input and output for each part is also shown.

makes it possible for the network to iteratively reevaluate higher-order spatial relationships to improve the prediction accuracy.

4. Proposed Detection Pipeline

As shown in Fig. 4, the detection pipeline consist of two main parts: Stacked Hourglass Network for detection of cuboid primitives by regression of heatmaps and the second part – decoder. The ConvNet output consists of multiple channels, each for a different type of corners and edges (see Fig. 5), where each channel contains a pixel-wise score of the key point. Another part is the decoder, which is responsible for extracting the parametric representation of cuboid primitives, connecting them in order to obtain the models of the boxes and correcting badly detected points based on cuboid geometry and other criteria where possible. Each of these processes and used algorithms will be addressed in more detail in the corresponding sections later in this article.

5. Dataset

Because there is no publicly available dataset of cardboard boxes, one of the first steps was to collect one.

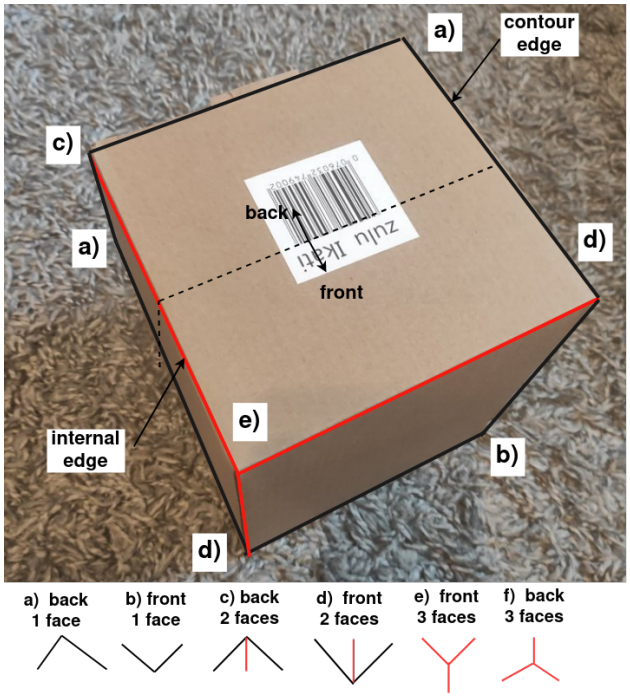


Figure 5. Different types of corners by the number of visible faces contributing to the corner and whether the corner is on the back side (faces back) or front side of the box (facing front). Two different edge types are also shown denoted by color (contour edges – black and internal edges – red).

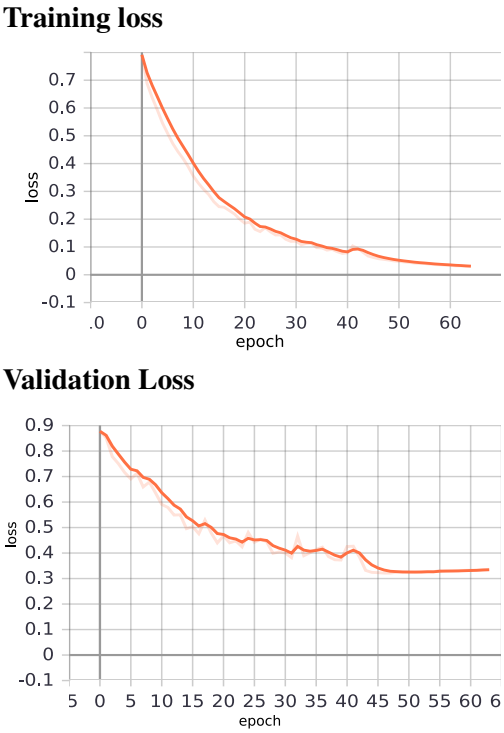


Figure 6. Graphs are showing the validation and training loss in epochs of training. We can see that since around epoch 50, only the training loss is improving while the validation loss started slowly diverging indicating over-fitting.

It consist of images containing single or multiple cardboard boxes of different colors and sizes. When it comes to the background, both simple single-colour background and more complicated ones such as patterned carpets are present. Photos were taken indoor under both natural and artificial lightning. Each image contains either one or more instances of the box. Images are annotated with corner and edge key-point coordinates and the corresponding class describing the type of the key-point. Currently, the dataset contains 536 images with 3102 annotated corners and 4044 edges and I am gradually adding more.

5.1 Corner and Edge Classes

In order to capture additional information ,which could be useful in finding out the orientation of the box or correcting errors in predictions, I have divided corners and edges into multiple classes. Edges are divided into 2 classes – internal and contour and corners have 6 types according to number of visible faces contributing to the corner (1, 2, or 3) and according to whether the corner is on the back side or front side of the box. Practical examples of these cases are shown in Fig. 5.

6. Training

The model is implemented using Python with TensorFlow [7] and the implementation is based on the Stacked Hourglass Network [6] inspired by a public implementation[1]. As an optimizer the Adam optimizer [8] with default TensorFlow 2 parameters was used and when it comes to loss function, I decided to use custom weighted euclidean loss (Section 6.2).

For training, the data were randomly split into training, validation and test splits using 80 : 10 : 10 ratios. The training was done on a single Nvidia GTX 1060 GPU for 65 epochs which took approximately 12 hours. The best validation loss was achieved after 48 epochs (Fig. 6).

6.1 Ground-Truth Data Synthesis

Because the annotation consists of key-point coordinates and class labels and the network outputs per-pixel key-point score in 8 channels (one per class), the ground truth annotations must be prepared for the training. This is done for each image by creating a $128 \times 128 \times 8$ tensor of zeros, where 8 represents the

[1] https://github.com/ethanyanjiali/deep-vision/tree/master/Hourglass/tensorflow

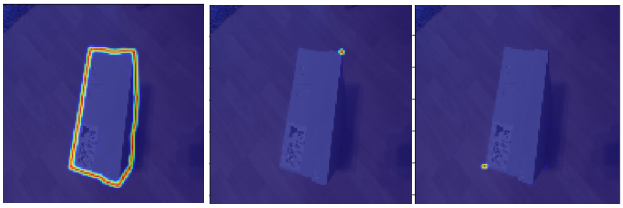


Figure 7. Example of ground truth data, showing 3 channels out of 8, one for external edge key-points and 2 for corners (single face facing back and single face facing front).

number of classes. Then, a 2D Gaussian is inserted for each key-point into a corresponding channel in a way that a peak is at a position of the key-point. The value of the σ is fixed and was chosen experimentally to be 1 for edges and 1.5 for corners, whereas increasing it from 1 to 1.5 for corners resulted in slightly better model accuracy. Increasing σ for edge key-points did not help, probably because each edge consist of multiple edge points close to each other, covering a bigger area than in the case of corners.

6.2 Loss Function

The loss function most commonly used for the regression task is Mean Square Error (MSE) and because of that, it was also our first choice. However, early stage experiments shown that the model had a hard time learning Gaussians, especially for corners, and the cause of the problem was tracked to the loss function. Because in this application, the ground truth data contain only a few non-zero values and a multitude of zeros, the network had the tendency to learn a trivial solution predicting all pixels as zeros and getting stuck in this local minimum of the loss function. To solve this problem we used a custom loss function using weighting of the standard MSE according to the ratio of non-zero pixels (denoted as n_b) and total pixels ($w \times h$) in ground truth. Weights are defined for each pixel in the prediction as follows

$$w_{ij} = \begin{cases} 1 - \frac{n_b}{w \cdot h} & \text{if } l_{i,j} > 0 \\ \frac{n_b}{w \cdot h} & \text{otherwise} \end{cases}, \qquad (1)$$

where i and j are indices of the corresponding pixel. Then, pixel-wise squared difference between the predictions y and labels l is weighted using the weights. Then, the loss is reduced to scalar by taking the mean value:

$$E = \frac{1}{w \cdot h} \sum_{i=1}^{w} \sum_{j=1}^{h} (y_{ij} - l_{ij})^2 \cdot w_{ij}. \qquad (2)$$

7. Processing an output of the ConvNet

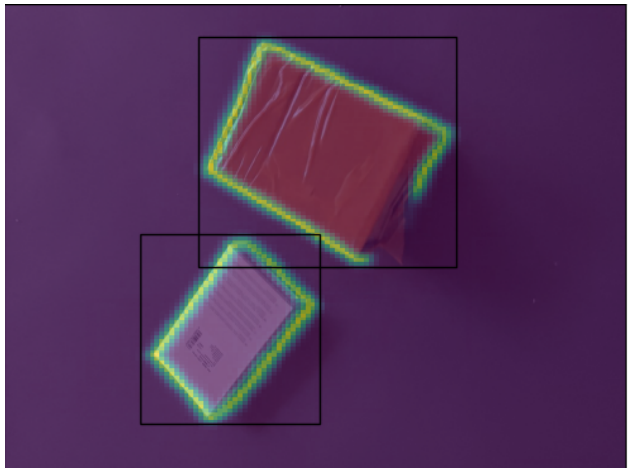


Figure 8. The image shows connected components in the predictions of outer edges and the estimation of the bounding boxes based on that.

7.1 Separating Instances of the Boxes

As the system uses bottom-up detection approach the network output contains key-points which may form multiple boxes and need to be assigned to the corresponding objects. In the first step, this is done by finding bounding boxes based on external edges.

Each box has multiple external edges which are connected and together form a polygonal area containing the box as shown in Fig. 8. Firstly, the system identifies connected components in the thresholded outer edge predictions. Given that the boxes do not overlap each other, each component corresponds to a single box and its bounding box can be obtained. Then, for every box described by the bounding box, every key-point outside of this area are masked before passing it to next processing step towards constructing the shape model of the box. As stated before, this method fails in cases where two or more boxes overlap each other, making their outer edges form a single connected component and this case must be solved later in the process.

7.2 From the Heatmap to Coordinates of Corners

The network produces an output consisting of 8 channels of per-pixel scores for each keypoint class. Based on that, it is necessary to get the opposite of the process of preparing the ground truth in Sec. 6.1 – to obtain the key-point coordinates. The first step will be applying a threshold (value around 0.2) to the output to suppress pixels with very small scores, in order to avoid false detections. After that, all separated non-zero regions are identified where each will represent a single Gaussian / an image of a corner. Then, x and y coordinates of the keypoint are obtained by computing the center of gravity of the region.

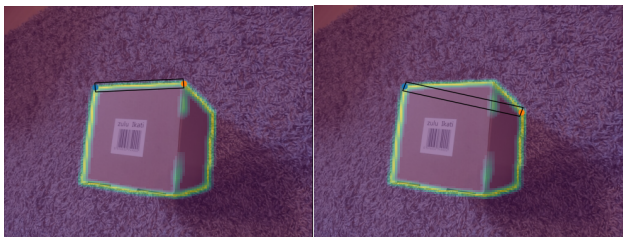


Figure 9. Example of correct an incorrect corner pair (91 % and 26 % of non-zero edge responses under the mask.

7.3 Finding Corner Pairs

The initial processing produces sufficient information to find the shape model – locations of the corner and edge key-points – but it must be assembled together and corner pairs corresponding to the edges of the box must be found. This process is based on the fact that if 2 corners forms an edge, there must be edge points on their connection. The algorithm works by taking corner pairs and finding if there is an edge on their connection. This is done by taking a rectangular mask over the connection with an arbitrary chosen width and computing the ratio of zero to non-zero responses of the edge-detector under this mask. If this ratio is bigger than a given threshold, the corners are considered to form an edge which corresponds to the edge points under the mask in the edge channel. Then the edge responses under the mask are set to zero and the process repeats until all edges are found.

7.4 Correction of Missing Corners

Up till now, the set of corner pairs each describing and edge of the box – the shape model – has been found. However, in some cases, some corners might be missing causing errors in the shape model. What can be done in some cases is to use a cuboid geometry to identify and correct these errors to a certain degree. A common example of this kind of error is when one corner was not detected, in this case probably because it was very close to another corner. Because of this, the shape model is incorrect in a way that one face of the box is forming a triangle instead of a rectangle as shown in Fig. 10, left. This can be identified and corrected by estimating location of the missing corners by completing the triangle into a parallelogram (Fig. 10, right).

8. Experimental Evaluation

8.1 Keypoint Localization Accuracy

To evaluate keypoint localisation accuracy, 2 metrics were used.The first metric is widely used in human pose estimation domain and is called PCK [9], which

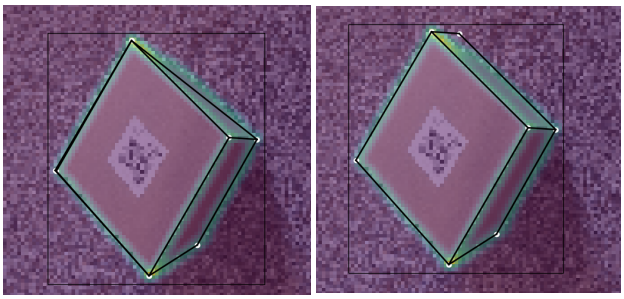


Figure 10. Example of estimated position of a missing corner by completing triangular face of the box into a parallelogram.

refers to probability of correct keypoint. The second one is recall of keypoint. Both metrics are evaluated using a defined relative distance threshold, from now denoted as α, which determines the maximal distance to the ground-truth keypoint relative to the detected part size for it to be considered as correct. In human pose estimation most often head size is used as reference. In our case we used diameter of the Gaussian in ground truth as a reference. Given P_a is the set of all ground truth keypoints and P_d is set of all detected keypoints the PCK is calculated as

$$PCK = \frac{|P_c|}{|P_a|} \cdot 100\,\%, \tag{3}$$

where P_c denotes set of correct detections (within distance α form ground-truth. The recall is then calculated as

$$R = \frac{|P_d|}{|P_a|} \cdot 100\,\%, \tag{4}$$

where P_d is set of detected ground-truth keypoints (with corresponding detection in P_c within distance α).

8.2 Bounding Box Accuracy

Although this system primary output is a shape model of the box also bounding box evaluation is used mostly to rate the boxes separation step. The decision whether the bounding box is correct is based on intersect over union (IoU), which is a standard way to do it. The process consist of calculating the intersect and union of ground truth and predicted bounding box and comparing the intersect to union ratio with a given threshold (most often 0.5). If it's bigger than the threshold, prediction is considered correct, otherwise it's considered wrong. The exact results are shown in Table 1.

8.3 Effect of Different Threshold Values

Because the CNN outputs per-pixel score for each key point class first processing step is tresholding the output to eliminate false positives. The choice of the

Recall @ 0.5 IoU	Acc. @ 0.5 IoU	# test imgs
96.82 %	76.25 %	54

Table 1. Bounding box evaluation of the first stage box separation. This separation is done by finding connected components in outer the edge heatmap as described in Sec. 7.1. In this step high recall is a priority because accuracy could be improved during next steps.

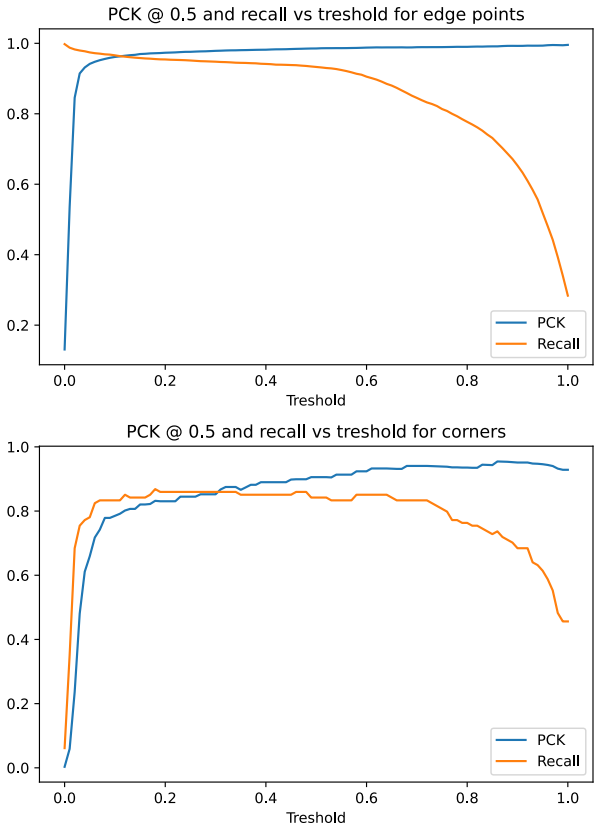


Figure 11. The graphs show the relationship between recall and probability of correct keypoint vs. threshold for both edge points and corners. Increasing the threshold leads to better accuracy as only those keypoints where the model was more confident were accounted for. On the other hand, increasing the threshold too much has a negative impact on the recall as also correct key-points are suppressed. The reason behind low recall for corners when threshold is close to zero is in the method of extracting coordinates from predictions (Sec. 7.2). In cases of very low threshold, multiple corners predictions form a single connected component. For both evaluations $\alpha = 0.5$ was used (Sec. 8.1).

threshold value have significant impact on the performance. The correlation between the threshold, PCK and recall can be seen in Fig. 11.

8.4 Effect of Dataset Size

In supervised learning, the dataset size is known to have a significant impact on the model performance. Because our system is trained on a self-made dataset, which I was gradually labeling during the creation of this work, this effect was very easy to notice. The main value of this information is that by looking on the exact numbers in Table 2, one can assume that there is still some hidden potential to improve the detector by enlarging still relatively small dataset.

Table 2. Dataset size vs. corners PCK

# train images	# corners	PCK @ 0.5	Recall @ 0.5
150	838	82 %	79 %
300	1706	87 %	85 %
436 (all)	2492	89 %	86 %

9. Conclusions

This paper presents an application of heatmap keypoint regression, which is a widely used method in human pose estimation in a different field – cardboard boxes detection. I proposed and implemented a detection system which takes RGB image as an input and outputs shape models of detected boxes. The system have achieved PCK 90 % and recall 86 % for corners and 97.5 % and 96 %, respectively for edge points and it can be used as a base for box detection implementation for automated systems in fields like transportation and packaging.

Based on observations of the dataset size and accuracy relationship, which is described in Section 8.4, the accuracy of the system could be further improved by using more complex augmentation methods, training on synthetic or simply larger datasets. Another idea for further work is to experiment with top-down approach detector by combining keypoint detection with standard region based detector for instance separation.

Acknowledgements

I would like to thank my supervisor prof. Ing. Adam Herout, Ph.D. for his valuable advice and support during the creation of this work.

References

[1] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[2] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look

once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[3] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[4] Debidatta Dwibedi, Tomasz Malisiewicz, Vijay Badrinarayanan, and Andrew Rabinovich. Deep cuboid detection: Beyond 2D bounding boxes. *CoRR*, abs/1611.10010, 2016.

[5] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2D human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[6] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked Hourglass Networks for human pose estimation. *CoRR*, abs/1603.06937, 2016.

[7] Paul Barham Martín Abadi, Ashish Agarwal et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[9] Tewodros Legesse Munea, Yalew Zelalem Jembre, and Halefom Tekle Weldegebriel et al. The progress of human pose estimation: A survey and taxonomy of models applied in 2d human pose estimation. *IEEE Access*, 8:133330–133348, 2020.