Detecting Cups and Line Boundaries of Pantry Furniture in a Tea-Making Scene

Anouk Bosmann
U1258920

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
DEPARTMENT OF COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE
SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
TILBURG UNIVERSITY

Thesis committee:

Dr. Sharon Ong
Dr. Marie Postma

Tilburg University
School of Humanities and Digital Sciences Department of Cognitive Science & Artificial Intelligence Tilburg, The Netherlands
June 2019

Preface

This thesis is written in the completion of my education at Tilburg University, resulting in a Master of Science degree in Data Science & Society. First, I would like to thank dr. Assistant Professor Sharon Ong for supervising and guiding me through the process of writing this thesis. Further, I would like to thank dr. Marie Postma for her feedback and time evaluating my work.

Detecting Cups and Line Boundaries of Pantry Furniture in a Tea-Making Scene

Anouk Bosmann

Object detection is an important task of computer vision. This research focuses on the detection of cups and shelves from a first-person view of a participant who is making tea. This data is acquired from a scene camera mounted on a pair of glasses that the participants wear. Making tea is a daily recurring activity for most people where a participant performed the task automatically, without thinking much (Land, Mennie, & Rusted, 1999). This task is often used for behavioral studies. Manual analysis to detect cups and lines representing the boundaries of the shelves in these videos is labor intensive. There is a need for automated techniques to analyze these videos. In the last couple of years, the performance of object detectors has improved. To demonstrate what improvements have been made, we compare the state-of-the-art object detector RetinaNet with the baseline classifier template matching. Furthermore, line detectors are analyzed to examine how well the Progressive Probabilistic Hough transform (PPHT) performs compared to the standard Hough transform at detecting the line boundaries of pantry furniture in the tea making scene. There is no ground truth solution available for our dataset, video annotation was necessary. This had led to the sub research question of could tracking approaches help the manual annotation of cups. A video is made where the cups are tracked and the boundary lines are detected. This video shows where the cups are in the environment. The results show that RetinaNet outperforms template matching and that the PPHT is better at detecting the shelves and cupboards than the standard Hough transform.

Contents

1. Introduction

Every time we open our eyes, we need to deal with a vast amount of information. We can easily filter important information from unimportant noise (Carrasco, 2011). However, this filtering can be a challenge for computers. The human brain has an extensive feedback system in the visual process, that is not fully understood yet. This feedback system uses the years of experience living in the world and inputs from all sorts of sensors, to suppress all irrelevant parts and to focus on the important areas of the image (Kaehler & Bradski, 2016). An image consists of a lot of information, but when a computer sees an image, all it 'sees' is a grid of numbers. These numbers consist of noise for a large part. The function of computer vision is to turn these numbers into something meaningful (Kaehler & Bradski, 2016).

There are many different application fields in daily life, where computer vision plays a role. For example, in the care of older adults with dementia. Mihailidis, Carmichael, and Boger (2004) discuss a sensing agent, that can automatically determine what kind of situation the elderly person is in and determine if support is needed. Another application is the development of autonomous vehicles (Janai, Güney, Behl, & Geiger, 2017). Furthermore, law enforcement also uses computer vision, for instance in facial recognition, security cameras, and license plate readers (Idrees, Shah, & Surette, 2018). One of the issues of computer vision is that there are too many images and videos in the world to process as humans. The number of images has increased enormously since the arrival of camera-equipped phones. If these images are not taken care of, a lot of information will be lost (Gumgum, 2016). To give a general idea about the number of images, Photoworld (2015) calculated that for a human being, it will take ten years, to see all the images that are shared on Snapchat in the last hour. The only way to be able to deal with this immense number of images is by the continuous development of computer vision techniques.

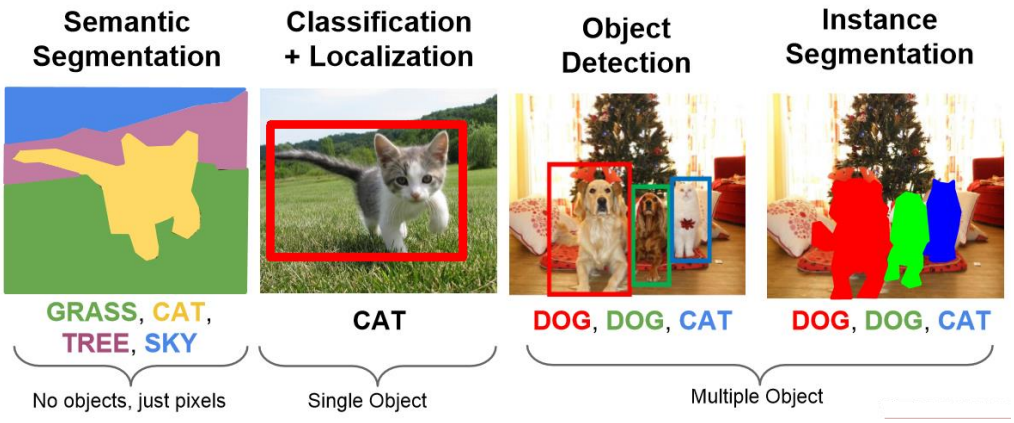

Figure 1 Other computer vision tasks. Reprinted from *GitHub* website, by F.F. Li, J. Johnson, and S. Yeung, 2017, retrieved from https://github.com/khanhnamle1994/computer-vision/blob/master/Lecture-11-Detection-and-Segmentation/cs231n_2017_lecture11.pdf

Computer vision can be divided into different tasks (Figure 1), whereas this thesis focuses on the task of object detection. The goal of object detection is to find items of interest in an image, draw a bounding box around the object and assign a label to the bounding box (Long, 2018). The results of object detection have improved rapidly over the last couple of years. The most attribution comes from Convolutional Neural Networks (CNN) (Jain, 2016; He, Gkioxari, Dollár, & Girshick, 2017). The goal of this thesis is to gain insight into the improvement of object detection. To achieve this, we compare a state-of-the-art CNN with a baseline. Template matching is used as the baseline because it is the simplest object detection technique. The algorithm finds the object in the image by simply sliding a template over the image and finding the location with the best match (Hashemi,

Aghdam, Ghiasi, & Fatemi, 2016). RetinaNet (Lin, Goyal, Girshick, He, & Dollár, 2017) is the state-of-the-art CNN. This method is selected because it surpasses all other state-of-the-art object detectors, e.g. R-FCN and Faster R-CNN, in accuracy and computation time (Lin, Goyal, et al., 2017).

The data are videos, from a study of Ioannidou, Hermens, and Hodgson (2016). These videos are made from the point of view of the participants, who performed multiple tasks. The task analyzed is making tea because making tea is a daily recurring activity for most people. The participants performed the task automatically, without thinking much (Land et al., 1999). This yields videos where the participants act the same as they would normally do. Because the data are close to reality, the results are usable in the real world. An example of this usage is that it can support blind people with making tea, in a for them unknown environment. A device could be developed that detect the objects of interest and tells the blind person where the objects are. Likewise, it could support elderly people with dementia with making tea.

The object detection in this thesis focuses on detecting cups. The reason for cups is that in the data there are multiple cups present. These cups differ from each other in shape and color. The videos captured the cups at different angles and lighting conditions. Furthermore, there is interaction between the participants and the cups. For example, the participants lift a cup from the shelf and tea is poured into it. This leads to the first research question:

> *"To what extent does RetinaNet improves cup detection in video recordings acquired from a participant's point of view, while they are making tea?"*

A subtopic of this study is to detect where the cup is in the context of the environment. The cups can be on the shelves, in the cupboards or on the kitchen counter. Knowing where a cup is in the environment is interesting because it gives information about the status of the cup. For example, if the cup is on the kitchen counter there is a higher chance that it contains hot water, then when it is in the cupboard. This is important information for the safety of blind persons when they are making tea in a for them unknown environment. In images, the shelves, counters, and cupboards are made up of long straight lines. There are multiple methods for detecting lines in an image. In this thesis, two line detectors are compared. The standard Hough line transform and the Progressive Probabilistic Hough Transform (PPHT). The reason for analyzing the standard Hough line transform is that it is a well-known common line detection algorithm (Zheng, Luo, Song, Yan, and Wang, 2018; Elsalamony, 2015; Seifozzakerini, Yau and Mao, 2017). PPHT is chosen because previous research shows that it is an improvement of the standard Hough transform (Matas, Galambos, & Kittler, 2000). Furthermore, both methods do not require labeled data. PPHT only examines a randomly selected subset of points and is, therefore, a faster line detector than the standard Hough line transform (Matas et al., 2000). This thesis is going to examine if PPHT also performs better at detecting lines in a tea making scene. This leads to the second research question:

> *"How well does PPHT performs compared to the standard Hough transform at detecting line boundaries of pantry furniture in the same video recordings?"*

There is no ground truth solution available for our dataset for us to answer our research questions. Manual annotation of each object in each video frame for ground truth is a labor-intensive process (Janai et al., 2017). As the annotation errors should be monitored, labeling ground truth in videos cannot be fully automated. The alternative to manual annotation are semi-automated approaches where the object of interest can be manually annotated in the first frame and tracking approaches can be applied to track and

label the object in subsequent frames. Incorrect annotations can then be manually corrected by the users. This has led to the sub research question which is:

"Could tracking approaches help the manual annotation of cups in our dataset?"

The remainder of this thesis is organized as follows. Section 2 contains a description of related work. Section 3 describes the experimental setup, dataset, and methods. Section 4 presents the results while Section 5 provides a discussion of the results. Section 6 presents my conclusions and recommendations for future research.

2. Related Work

This section presents an overview of dataset annotation, some background information on images and related work in line detection and object detection.

2.1 Dataset Annotation

Ten years ago, a dataset with a few hundred annotated instances was treated as acceptable because gathering massive amounts of labeled data is a challenge. The collection of the data itself is not the problem, but it is labor intensive and expensive to accurately annotate the data with ground truth labels (Vondrick, Ramanan, & Patterson, 2010). This costs a lot of effort because it requires manual labeling (Janai et al., 2017). Fully automated annotation is not possible as the annotation errors should be monitored. There has been an effort to semi-automate the annotation by supervised learning (Janai et al., 2017). The simple approach of linear interpolation was explored (Vondrick et al., 2010). With this approach users annotate every nth frame with bounding boxes. They use linear interpolating to estimate the path of the object in the not annotated frames. A drawback of this approach is that it is only applicable if the object of interest moves linearly through the frames (Vondrick et al., 2010). Another approach is video annotation with interactive tracking (Vondrick & Ramanan, 2011; Buchanan & Fitzgibbon, 2011). Agarwala, Hertzmann, Salesin, and Seitz (2004) and Yuen, Russell, Liu, and Torralba (2009) both proposed video annotation were most of the work is automated and users only adjust the annotation errors of drifting trackers. Herein is most of the work automated. Today, there are many different annotation software available. The software differ from each other in functions, ease of use and price (Humans in the loop, 2018). Some of the annotation software offer a lot of tools, like polygons, dots, and circles. For this study, complicated software is not required. A tool that supports bounding boxes is sufficient. We chose to use the open-source annotation tool OpenLabeling (Cartucho, 2018). This is a simple and free tool that supports the functions needed.

2.2 Images

An image is a tensor of pixel values, where a tensor is an N-dimensional vector, with no theoretical maximum for the dimensions (Olafenwa & Olafenwa, 2018b). Pixels are the smallest components in an image. For an 8-bit image, the value of each pixel ranges from 0 till 255 and corresponds to the light intensity at a specific point (Olafenwa & Olafenwa, 2018b). For image analysis, the only information accessible for the algorithm is the pixel value (Olafenwa & Olafenwa, 2018b). Videos are a series of images over time. Therefore, the dataset of videos comprises of pixel values in a matrix.

There are two types of images; grayscale and color. Grayscale images are represented by a two-dimensional matrix where the image width and height correspond to the number of columns and rows in the matrix. The matrix consists of various shades of black, white, and gray (Olafenwa & Olafenwa, 2018b). Where a pixel value of 0 represents black and a value of 255 represents white (Figure 2). RGB (RED, GREEN, BLUE) or color images are represented by a three-dimensional matrix. The first two dimensions represent the width and height of the images while the third dimension represents the specific light wavelength; red, green, and blue (Olafenwa & Olafenwa, 2018b) (Figure 3 and Figure 4). Here RED stands for the intensity of red light, GREEN for the intensity of green light and BLUE for the intensity of blue light. The combinations of the colors RED, GREEN, and BLUE can form any color (Figure 5). The intensity of RED, GREEN, and BLUE light at a point determines the color of that point (Olafenwa & Olafenwa, 2018b).

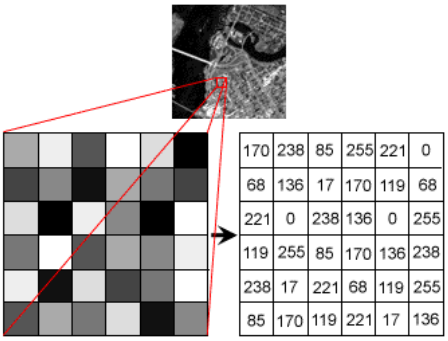

Figure 2 [Pixel values grayscale image]. Reprinted from *Introduction to Computer Vision* website, by AI Stanford, n.d., retrieved from http://ai.stanford.edu/~syyeung/cvweb/tutorial1.html

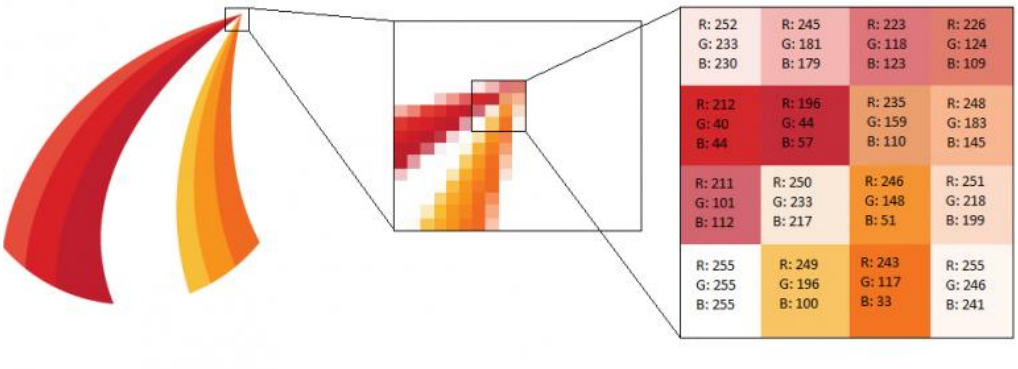

Figure 3 [RGB values of a pixel in an image]. Reprinted from *Analytics India* website, by K. Maladkar, 2018, retrieved from https://www.analyticsindiamag.com/computer-vision-primer-how-ai-sees-an-image/

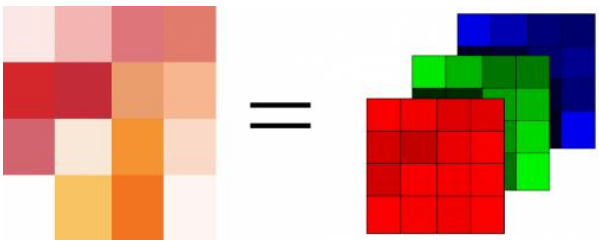

Figure 4 [Three color channels placed behind each other]. Reprinted from *Analytics India* website, by K. Maladkar, 2018, retrieved from https://www.analyticsindiamag.com/computer-vision-primer-how-ai-sees-an-image/

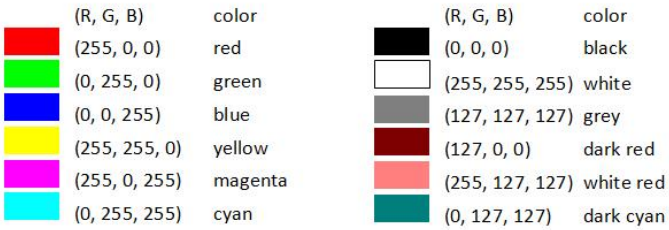

Figure 5 [Color examples with RBG values]. Reprinted from *Dia scan* website, by Dia Scan, 2013, retrieved from http://dia-scan.blogspot.com/2013/02/digitized-picture-correction-with-digiKam-part2.html

2.3. Line Detection

An application of computer vision is line detection, wherein the goal is to find line segments in an image. The motivation for detecting lines is, that they represent intensity discontinuity. These discontinuities are the basis for detecting objects and surface boundaries (Von Gioi, Jakubowicz, Morel, & Randall, 2010; Barrow & Tenenbaum, 1981). Classic approaches for line detection apply a Canny edge detection followed by the Hough line transform (Von Gioi et al., 2010; Suárez, Muñoz, Buenaposada, & Baumela, 2018). Although the Hough line transform is a useful algorithm to detect straight lines in an image, this algorithm has several disadvantages (Ali et al., 2018). Areas with high edge detection can lead to false detection and the orientation of each edge is not considered. Furthermore, the algorithm requires the user to set a fixed threshold that distinguishes whether a line exists, which can lead to a high number of false positives or false negatives (Von Gioi et al., 2010). The PPHT (Matas et al., 2000) is an improvement over the standard Hough transform. PPHT performs better at detecting lines and minimizes the amount of computation needed (Von Gioi et al., 2010). The computation time of PPHT is less because only a randomly selected subset of points is examined, without affecting detection accuracy (Matas et al., 2000). However, PPHT has some drawbacks. It does not detect small line segments well and it is dependable on the image size. There is no related work that specifically looks to detect line boundaries of pantry furniture. However, there is a lot of related work using Hough transforms. For example, to detect roads and lane makings from videos (Li et al., 2016) and to detect the corridor lines in an indoor environment (Shi & Samarabandu, 2006). This thesis contributes to the current work to specifically look to line detection of pantry furniture.

2.4 Object Detection

Every time humans open their eyes, they unconsciously detect objects. For a computer to do the same when the camera is moving, there are some challenges to overcome. A challenge is that an object can appear radically dissimilar from different viewpoints (Figure 6) (Kaehler & Bradski, 2016). Other challenges are if the object of interest is partly covered or if it has blended into the background (Jain, 2016).

A classic and fundamental method for object detection is template matching. There has been a lot of research for template matching (Brunelli, 2009). Template matching tries to find small regions in an image which match the template image of the desired object (Hashemi et al., 2016). Different template matching methods have been implemented successfully over the years (Oron, Dekel, Xue, Freeman, & Avidan, 2018). There are some disadvantages to template matching. Although template matching is simple to implement, it is may return poor results due to illumination, background, and scale changes (Oron et al., 2018). Furthermore, template matching may also fail if there is a lot of noise in the image or if the object of interest is occluded (Elboher & Werman, 2013).

Another object detector is CNN (Janai et al., 2017). The original CNNs used a sliding window approach (Sermanet, Kavukcuoglu, Chintala, & LeCun, 2013), but these CNNs suffer from a localization problem (Janai, et al., 2017). To overcome this problem Region-based CNNs (R-CNN) were proposed (Girshick, Donahue, Darrell, & Malik, 2014). However, R-CNNs are computationally expensive (Janai et al., 2017). The introduction of Fast R-CNN (Girshick, 2015) and Faster R-CNN (Ren, He, Girshick, & Sun, 2015) has reduced running time with the use of Region Proposal Networks (RPN). Dai, Li, He, and Sun (2016) presented a Region-based fully Convolutional Network (R-FCN) which achieves the same competitive results as Faster R-CNN but is several times faster in computation time. In this thesis, the detector RetinaNet will be analyzed, because it surpasses state-of-the-art detectors Faster R-CNN and R-FCN in accuracy and computation time (Lin, Goyal, et al., 2017). RetinaNet performs better than the other object detectors

because it has solved the problem of class imbalance. This problem is solved with the use of Focal Loss. The Focal Loss is a reshaped cross entropy loss (Lin, Goyal, et al., 2017). A detailed description of RetinaNet is given in section 3.3.2. There is no related work specifically for detecting cups in videos from a first-person view, though there is related work which uses CNN to detect other objects in indoor environments. Hernández, Gómez, Crespo, and Barber (2016) detected objects, e.g. chairs and closets, from a camera mounted on a moving robot. Adhikari, Peltomaki, Puura, and Huttunen (2018) detected fire extinguishers, chairs, exit signs, clocks, trash bins, screens, and printers in indoor scenes. This thesis contributes to the current work to specifically look to cup detection in videos from the first-person view.

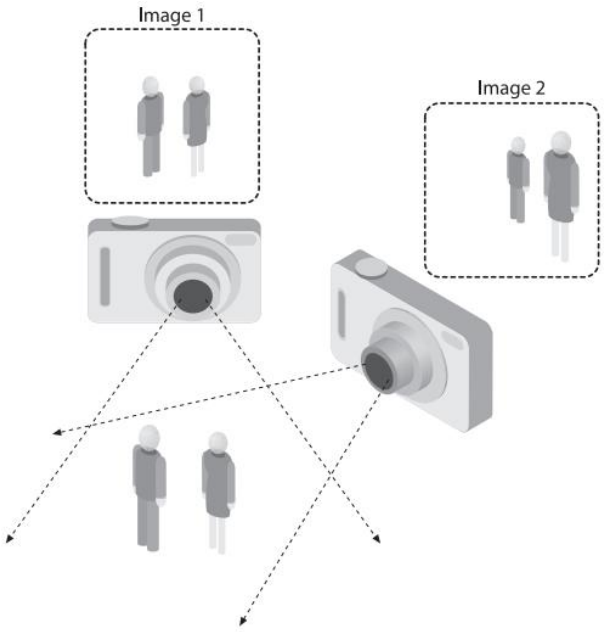

Figure 6 The ill-posed nature of vision: the 2D appearance of objects can change radically with viewpoint. Reprinted from *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library* (p. 4), by A. Kaehler & G. Bradski, 2016, California, CA: O'Reilly Media

3. Experimental Setup

This section contains a description of the experimental setup. First, we present the data that is used. Second, we present the annotation of the data with the use of a tracker. In the last two parts, we give descriptions of the object detection models and the line detection models.

3.1 Data

The dataset used for this study are videos, in the first-person view, collected in a study by Ioannidou et al. (2016). This study collected data from 42 participants (Males = 14, Females = 28, aged from 18 to 46 years old; mean = 21.38, SD = 5.18). The Tobii Pro 2 ultra-light head-mounted eye tracker (Figure 7) recorded the data. The eye tracker contains a pair of glasses and a recording unit. The eye tracker had eye cameras that recorded the eye movements as well as a scene camera on the outside of the glasses. The scene camera recorded video images from the participant's point of view. The resolution of the scene camera is 1920 by 1080 pixels. Which corresponds to a field of view of 52 degrees vertically and 82 degrees horizontally. The data analyzed in this study are the video images (.mp4) from the scene camera. The data of the eye movements were not used in this study.

All the participants completed three tasks: a navigation task, a tea making task and a card sorting task. Only the part of the day to day task, tea making, was used in this study. The tea making task was inspired by an experiment by Land et al. (1999). The task was carried out inside a kitchen. Before the participants were led to the kitchen, they were explained that they were to make a cup of tea for the experimenter. In the kitchen, they got additional information. They were told that they needed specific items to complete the task. These items were placed in the cupboards in the kitchen (see Figure 8). The specific items were a green jar with tea, a red jar with sugar, a spoon placed in the front of one of the cupboards, a cup with colored butterflies, and a small bottle of milk that was placed inside the fridge. To strengthen the feeling of a real kitchen, none of the other items in the kitchen were removed. The task-relevant items were placed in the same locations for all the participants. The participants were reminded to act as naturally as possible, that there was no time limit to the task and that they were free to search all the cupboards in the kitchen.

Figure 7 The Tobii 2 glasses system. Reprinted from "The centrial bias in day-to-day viewing," by F. Ioannidou, F. Hermens & T.L. Hodgson, 2016, *Journal of Eye Movement Research, 9*, p. 4

Figure 8 Example cupboard for tea making. Note the target cup in the middle of the cupboard. Reprinted from "The centrial bias in day-to-day viewing," by F. Ioannidou, F. Hermens & T.L. Hodgson, 2016, *Journal of Eye Movement Research, 9,* p. 4

The duration of the videos is +/- nine minutes and the size is +/- 329 MB. The tea making part in the video is about 2.20 minutes in duration. In this thesis, the video of one participant was analyzed. All preprocessing is done with the use of the library OpenCV (Open Source Computer Vision Library) version 4.0.0 (OpenCV, 2018). OpenCV is a library consisting of more than a hundred computer vision algorithms. First, the video was taken from a file using VideoCapture(). The video consisted of 13487 frames with 25 frames per second. All frames in the video had a size of 1920 x 1080 (width x height). Second, VideoCapture.read() was used to capture the data from the video and return the grabbed frame. The grabbed frames were saved as an image (.jpg) with the use of cv2.imwrite(). Because this thesis was focused on the tea making task all the frames that were about the navigation task or the card sorting task were deleted. The remaining tea making part consists of 3401 frames. For every implementation, in this study, is the programming language python version 3.6.5 used.

3.2 Annotation

In order to acquire ground truth data, a semi-supervised approach is used to annotate the locations of cups in all the frames. This approach combines manual labeling and object tracking over subsequent frames. In the first frame which each cup appears in the video, a tight bounding box was drawn around it. This bounding box around the cup over subsequent frames was automatically found through tracking algorithms provided by the OpenCV tracker software package (Cartucho, 2018). The tracked bounding box region is adjusted manually when required when the tracker failed to identify the cups. The OpenCV tracker will be explained in the next subsection. The bounding boxes were saved in YOLO format, as a text file, and in PASCAL_VOC format, as an XML file. The YOLO format had the normalized values of x center, y center, x width and y height, and the assigned label to it. The XML files had an order of class, x min, y min, x max, and y max coordinates of the bounding box. The output of the tracking solution was stored in a JSON file.

3.2.1 OpenCV Tracker

Tracking means localizing an object in consecutive frames of a video. Compared to detecting an object through standard object detection methods, tracking requires less computation time (Mallick, 2017). The reason is that with tracking you have prior information about the appearance of the object. If the object was detected in the previous frame, that information can be used to forecast the location in the next frame. OpenCV Tracker uses all the information it has so far about the object and performs a small search around the expected location. In contrast, with object detection, the algorithm has no previous knowledge about the location of the object, so it must start from scratch (Mallick, 2017). Thus, OpenCV tracker profits from the additional information. A tracker can lose track of an object. This can happen because the intended object is no longer in the image, or when it moves too fast for the tracker. Another common fall back of OpenCV tracker is that the bounding box may slowly slide away from the object (Mallick, 2017).

The goal of tracking is to find the object of interest in the current frame, given the information about the object in the previous frames. This information is stored by mathematical functions in the motion model. This model has information about the velocity and location of the object in the previous frames. Further, there is information about how the object looks in the previous frame, stored as parameters in an appearance model. The two models are used in combination to accurately predict the location of the object. First, the motion model predicts the location. Next, the appearance model is used in the neighboring area to improve the prediction of the location, based on appearance (Mallick, 2017). OpenCV tracker has eight different trackers available BOOSTING, MIL, KCF, TLD, MEDIANFLOW, GOTURN, MOSSE, and CSRT. Kernelized Correlation Filter (KCF) is the tracker used in this thesis because KCF scores well on accuracy and speed (George, Jose, & Mathew, 2018).

3.3 Object Detection Models

One task of this thesis was to examine to what extent RetinaNet improves cup detection. In the next parts, the different algorithms considered will be described, and it will be defined how the parameters and evaluation metrics were chosen. Used packages and libraries will also be discussed per algorithm.

3.3.1 Template Matching

Template matching is a high-level machine vision method to find parts of an image that are similar to the template image. It consists of two components the input image and the template image, where it is expected to find the template in the input image (Perveen, Kumar, & Bhardwaj, 2013). The template simply slides (moving one pixel at a time) over the input image and the algorithm compares at every point the pixels of the template with the pixels of the area of the input image at the place of the template (OpenCV, 2013). At every location, a metric is computed, and all these metrics are stored in a result matrix. This metric is a comparison result. The resulting matrix is a grayscale image, in which each pixel implies how much the pixel's neighborhood equals with the template. In Figure 9 an example of a result matrix is shown. The brightest location implies the highest match (the red circle). This location is the left top corner of the match between the template and the input image. A bounding box is drawn at this location, where the width and height are equal to the template (OpenCV, 2013).

Template matching is implemented with the use of OpenCV, with the function matchTemplate(). A parameter of matchTemplate() is method. OpenCV has six different methods for comparing the template with the input image. These six different methods were tested on five different images. The method CV_TM_CCOEFF had the best results, therefore this method was used. The formula of this comparison method is

$$R(x,y) = \sum_{x`,y`} (T`(x`,y`) \cdot I`(x+x`,y+y`))$$

where $T`(x`+y`) = T(x`,y`) - \frac{1}{w \cdot h} \cdot \sum_{x``,y``} T(x``,y``)$

$$I`(x+x`,y+y`) = I(x+x`,y+y`) - \frac{1}{w \cdot h} \cdot \sum_{x``,y``} I(x+x``,y+y``)$$

(*I* denotes the input image, *T* the template, *R* is the result). The summation is done over the image and/or the template patch: $x` = 0 \ldots w - 1$, $y` = 0 \ldots h - 1$. The template has a size of *w* x *h* (OpenCV, 2013). A disadvantage of template matching is that it does not examine rotations or scaling of the template. The evaluation metrics of template matching are precision, recall, and F1-score. Accuracy is not used as a metric because we do not determine true negatives. Since every pixel that is not a cup and is not detected as a cup would be considered as true negative. Hence, the classification result will be imbalanced as the true negatives would skew the result.

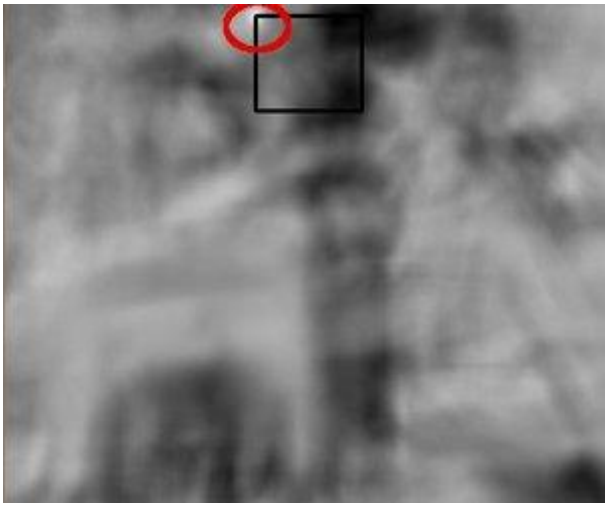

Figure 9 [Result matrix of Template matching]. Reprinted from *OpenCV* website, by OpenCV, 2013, retrieved from https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html

3.3.2 RetinaNet

In this thesis, RetinaNet was used for object detection (Olafenwa & Olafenwa, 2018a). RetinaNet is a single, unified network designed and trained by Facebook AI Research (FAIR) (Lin, Goyal, et al., 2017). RetinaNet consists of a backbone network and two subnetworks. The backbone is in charge of computing a convolutional feature map over a whole image. This is an off-the-shelf convolutional network. One of the subnetworks performs convolutional object classification over the output of the backbone. The other subnetwork performs convolutional bounding box regression over the output.

RetinaNet can detect 80 different objects, including cups. The detectCustomObjectsFromImage() function was used to detect only the cups in the images. The parameter minimum_percentage_probability was set at its default 50. The function returns a dictionary with the object name and the percentage probability of the detection. Each cup that was detected was saved as a separated image. The evaluation metrics used are precision, recall, and F1-score. Accuracy is not used as a metric for the same reason as mentioned above. The large number of true negatives will skew the results. In the next sections, a detailed description of RetinaNet is given.

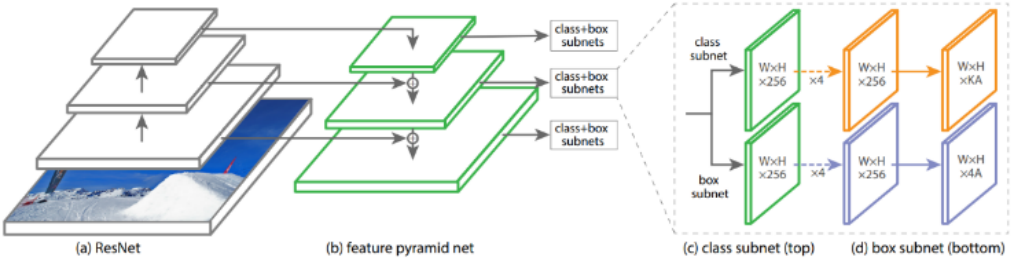

Figure 10 The one-stage RetinaNet network architecture uses a Feature Pyramid Network (FPN) (Lin, Dollár, et al., 2017) backbone on top of a feedforward ResNet architecture (He, Zhang, Ren, & Sun, 2016) (a) to generate a rich, multi-scale convolutional feature pyramid (b). To this backbone RetinaNet attaches two subnetworks, one for classifying anchor boxes (c) and one for regressing from anchor boxes to ground-truth object boxes (d). The network design is intentionally simple, which enables this work to focus on a novel focal loss function that eliminates the accuracy gap between our one-stage detector and state-of-the-art two-stage detectors like Faster R-CNN with FPN (Lin, Dollár, et al., 2017) while running at faster speeds. Reprinted from "Focal Loss for Dense Object Detection," by T.Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, 2017, *Proceedings of the IEEE international conference on computer vision*, p. 2984.

3.3.2.1 Feature Pyramid Network

The Feature Pyramid Network (FPN) is the backbone of RetinaNet (Lin, Goyal, et al., 2017). FPN improves a standard convolutional network through a top-down pathway and lateral connections. In this way, the network smoothly builds a rich, multi-scale feature pyramid from one individual input image (see Figure 10a-b). The FPN has two pathways, bottom-up and top-down. These pathways are connected through lateral connections. The bottom-up pathway is illustrated in Figure 10a. This pathway chooses the last feature map, from a group of successive layers. These consecutive layers all have output from feature maps on the same scale. The feature maps that are chosen will be used as the base of the feature pyramid (Lin, Dollár, et al., 2017). The top-down pathway with lateral connections is illustrated in Figure 10b. In the top-down pathway, the last feature map, from the bottom-up way, is increased to the same proportion as the second-to-last feature map by nearest neighbor unsampling. After this, the two feature maps are joined using an element-wise addition. Before merging the feature maps, the bottom-up pathway is subjected to a 1 by 1 convolution to decrease the channel dimensions. This process is repeated until each feature map, from the bottom-up pathway, has an equivalent new feature map. These feature maps are connected with lateral connections (Lin, Dollár, et al., 2017). The top-down pathway and lateral connections do not demand a lot of extra computations, but the results are spatially and semantically strong. This is better than the bottom-up pathway, which is semantical strong, but spatial coarse.

Each pyramid level can be used for detecting objects at various scales. FPN performs, on multi-scale predictions, better than fully convolutional networks (FCN) (Long, Shelhamer, & Darrell, 2015) and two-stage detectors, like Mask R-CNN (He et al., 2017) and Fast R-CNN (Girshick, 2015) (Lin, Goyal, et al., 2017).

Like Lin, Dollár, et al. (2017), the FPN is used on top of ResNet, in a fully convolutional fashion (He, Zhang, Ren, & Sun, 2016). This fully convolutional way makes it possible for the network to work with every image size as input and create proportionally sized feature maps, at numerous pyramid feature levels, as output. Higher level feature maps consist of grid cells that cover bigger areas of an image. These higher level feature maps are therefore more suited to detect larger objects. Vice versa, lower level feature maps contain smaller grid cells and are therefore more suitable to detect smaller objects (Figure 11).

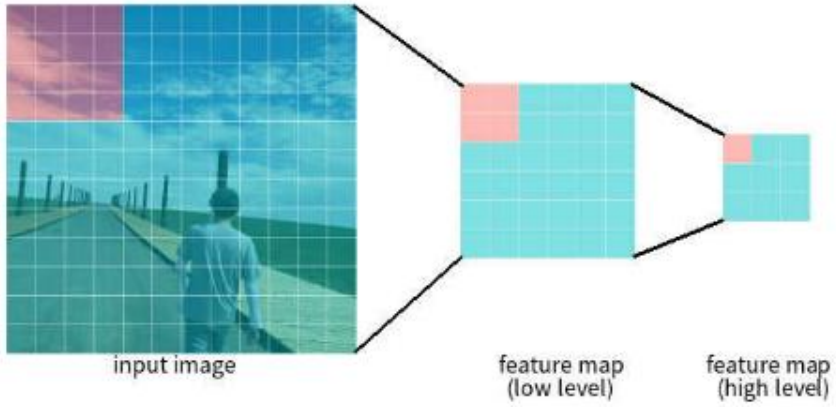

Figure 11 [Illustrating of high and low level feature maps]. Reprinted from *Blog Zeng* website, by N. Zeng, 2018, retrieved from https://blog.zenggyu.com/en/post/2018-12-05/retinanet-explained-and-demystified/

Seven pyramid levels are constructed. l expresses the pyramid level, where P_l has a resolution of 2^l lower than the input. All the levels of the pyramid have $C = 256$ channels. RetinaNet makes use of the feature level P_3 to P_5. These levels are computed from the output from ResNet, on the corresponding residual stage, C_3 through C_5. This is computed using lateral and top-down connections, following Lin, Dollár, et al. (2017). P_6 is computed through a 3x3 stride-2 conv on C_5. P_7 is obtained by using ReLu followed by a 3x3 stride-2 conv on P_6. Specifics of the pyramid mostly follow Lin, Dollár, et al. (2017), with some small differences. These minor modifications are that first, P_2, the high-resolution level, is not used due to computational reasons. Second, P_6 is obtained by stride convolution rather than downsampling. Third, P_7 is included to enhance large object detection. All these differences improve speed even though the accuracy is maintained.

3.3.2.2 Anchors

Anchor boxes are responsible for detecting objects and the shape and size of the objects. Anchor boxes were first proposed by Ren et al. (2015). Each anchor box covers an area of the input image and has a different size and aspect ratio. In Figure 12 are the anchor boxes, in black, visualized in an input image. The anchor boxes used have areas of $\{32^2, 64^2, 128^2, 256^2, 512^2\}$ pixels, on pyramid levels $\{P_3, P_4, P_5, P_6, P_7\}$, respectively. Anchors with three

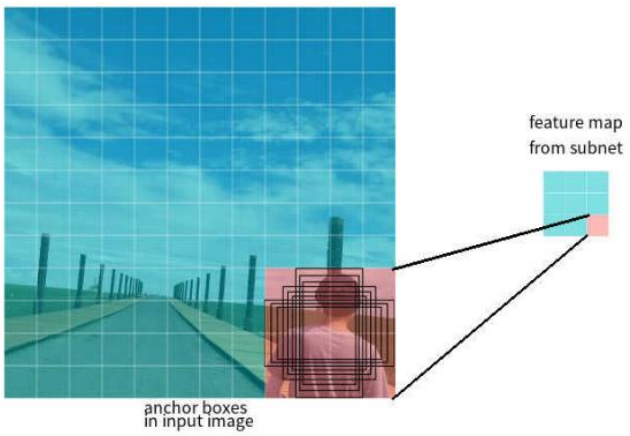

Figure 12 [Illustrating anchor boxes on input image]. Reprinted from *Blog Zeng* website, by N. Zeng, 2018, retrieved from https://blog.zenggyu.com/en/post/2018-12-05/retinanet-explained-and-demystified/

aspect ratios are used {1:2, 1:1, 2:1}. These are used at each pyramid level (Lin, Goyal, et al., 2017). This is similar to the anchors used by Lin, Dollár, et al. (2017). Different to Lin, Dollár, et al. (2017) is that to ensure denser scale coverage, anchors of sizes $\{2^0, 2^{1/3}, 2^{2/3}\}$ are added to the aspect ratios. In total there are $A = 9$ anchors at every level. These anchors cover, across levels, the scale range of 32-813 pixels, taking the input image into account. In Figure 13 the nine different anchor sizes, with different aspect ratios are visualized.

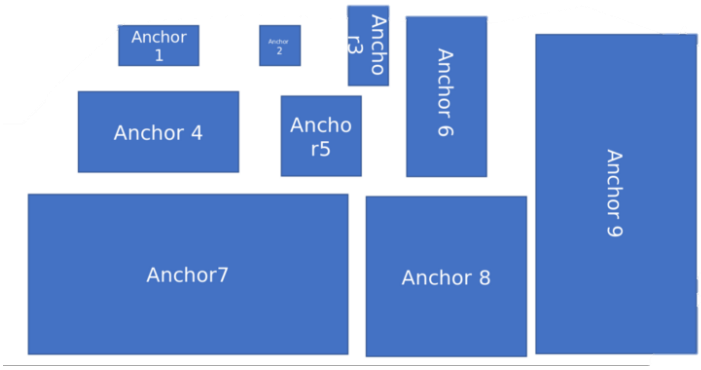

Figure 13 Anchor boxes mapping to Image from feature map. Adapted from *Medium* website, by P. Jay, 2018, retrieved from https://medium.com/@14prakash/the-intuition-behind-retinanet-eb636755607d

Each anchor is responsible for detecting objects from *K* classes, in the area of the input image, that it covers. *K* is the number of object classes. Each anchor has a *K* length one-hot vector of classification targets assigned. Each anchor is also responsible for detecting the shape and size of the object. Therefore, each anchor also has a 4-vector of box regression targets assigned (Lin, Goyal et al., 2017). An anchor box is assigned to a ground-truth box if the intersection-over-union (IoU) is greater than 0.5. In the case of classification, the corresponding entry in the *K* length vector is set to 1, while all other class entries will be set to 0. If the IoU is below 0.4 the anchor box is considered as background. In this case, all the class entries in the *K* length vector will be set to 0. If the IoU is between 0.4 and 0.5 the anchor box is considered to have no match with a ground-truth label and will be ignored during training. For box regression, the targets are computed as the offset, in terms of width, height, and center coordinates, between the anchor and the ground-truth box (Lin, Goyal, et al., 2017).

3.3.2.3 Classification Subnet

The classification subnet predicts, at each position, for each anchor and object class, the probability that an object is present. The classification subnet is an FCN, which is attached to each FPN level. The parameters of the FCN are shared across the FPN levels. The

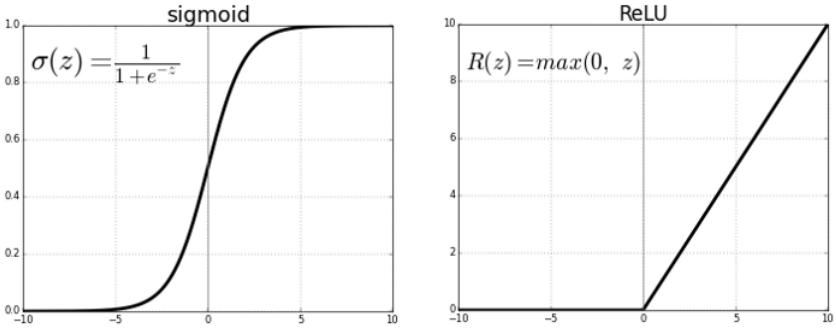

Figure 14 ReLu v/x/ Logistic Sigmoid. Reprinted from *Towards Data Science* website, by S. Sharma, 2017, retrieved from https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

classification subnet has four 3 x 3 convolutional layers with 256 filters. These are all followed by ReLu activations (Figure 14). After this, the subnet applies one 3 x 3 convolutional layer with *K* x *A* filters, followed by sigmoid activation (see Figure 14). The output feature map has a shape of *W*, *H*, *KA*. *W* and *H* are the width and height of the input feature map. The classification subnet is visualized in Figure 10c (Lin, Goyal, et al., 2017).

3.3.2.4 Box Regression Subnet

The regression subnet is also attached to each pyramid level of the FPN, similar to the classification subnet. The only difference between the classification subnet and the box regression subnet is that the last convolutional layer is 3 x 3 with 4*A* filters, instead of *K* x *A* filters. Therefore, is the shape of the feature output map *W*, *H*, 4*A*. Lin, Goyal, et al. (2017) use a class-agnostic bounding box regressor, this is unlike most recent work (like R-CNN and Fast R-CNN). A class-agnostic bounding box regressor detects objects without knowing to which class they belong to. Class-agnostic is the opposite of class-aware, where every bounding box has the class of the object associated with it (Huynh, 2017). The class-agnostic bounding box regressor has fewer parameters, but they found it to be evenly effective (Lin, Goyal, et al., 2017). The box regression subnet is visualized in Figure 10d.

3.3.2.5 The Loss Function

Object detectors that obtain the highest accuracy are two-stage detectors (Lin, Goyal, et al., 2017). Two-stage detectors apply a classifier on a sparse set of candidate object locations. Opposed to two-stage detectors are one-stage detectors. One-stage detectors apply a classifier for a dense set of candidate object locations. One-stage detectors are simpler and faster than two-stage detectors, but lag on accuracy. The central cause for this difference is the severe foreground-background class imbalance that is encountered during the training of the one-stage detectors. This class imbalance occurs because most locations in an image are easily classified as background by a detector, they are easy negatives (Lin, Goyal, et al., 2017).

RetinaNet is a simple dense one-stage detector, with the same speed as other one-stage detectors, while outperforming the accuracy of two-stage detectors. Lin, Goyal, et al. (2017) have achieved this by tackling the problem of class imbalance. They did this by adjusting the standard cross entropy loss in such a way that it down-weights the loss given to well-classified examples. This loss is known as the Focal Loss. The focal loss focusses on the few interesting examples. These examples contribute more to the loss than the many examples, that the network is already very sure about. Because it focuses on the hard examples the detector is not overwhelmed by the immense number of easy negatives (Lin, Goyal, et al., 2017). In Figure 15 the cross entropy and focal loss are visualized. In this figure, you can see that when the network is quite sure about a prediction, the focal loss is considerably lower than the cross entropy loss.

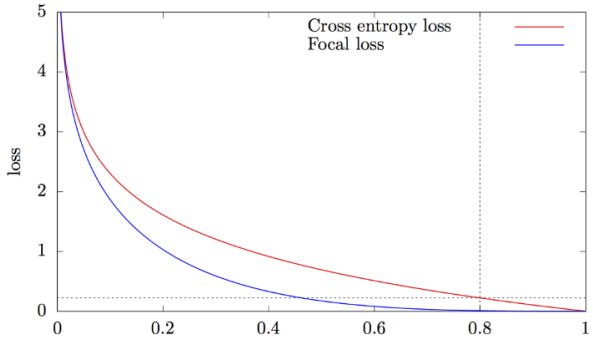

Figure 15 [Plot of cross entropy loss and focal loss]. Reprinted from *Towards Data Science* website, by F.M. Graetz, 2018, retrieved from https://towardsdatascience.com/retinanet-how-focal-loss-fixes-single-shot-detection-cb320e3bb0de

3.4 Line Detection Models

Another task of this study was to examine how well PPHT performs compared to the standard Hough transform to detect the line boundaries of pantry furniture. In the next parts, the different algorithms will be described, it will be defined how the parameters and evaluation metrics were chosen. Used packages and libraries will also be discussed per algorithm.

3.4.1 Hough Line Transform

The goal of the Hough line transform is to find the location of lines in an image. In Figure 16 the pipeline overview of the Hough line transform is shown. The input of the Hough line transform is a binary image. The original, color image is therefore converted to grayscale. After this, the edges are detected using Canny edge detection (Bradski & Kaehler, 2008).

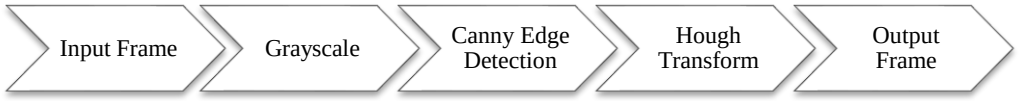

Figure 16 Pipeline overview of Hough Line Transform

3.4.1.1 Canny Edge Detection

Canny edge detection is implemented with the use of the OpenCV library. Canny edge detection was developed by John F. Canny in 1986. It is a multi-stage edge detection algorithm with four main stages (OpenCV, 2015).

(1) Noise reduction. Noise can be an issue with edge detection as it can lead to false detection. To remove the noise the image is smoothed with a Gaussian filter. This Gaussian filter applied has a 5 x 5 kernel of normally distributed numbers. This filter runs across the whole image and sets every pixel value to the weighted average of the neighboring pixels (Lin, 2018).

(2) Intensity gradient. The smoothened image is then filtered with a Sobel kernel, to get the first derivate, in horizontal (G_x) and vertical (G_y) direction. The edge gradient and the direction can be calculated as follows:

$$Edge\ Gradient\ (G) = \sqrt{G_x^2} + \sqrt{G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}(\frac{G_y}{G_x})$$

Where the direction is perpendicular to edges and will be in a horizontal, vertical or diagonal direction (OpenCV, 2015).

(3) Non-maximum suppression. To remove any unwanted pixels to sharpen the edges, non-maximum suppression is applied. For each pixel in the image, it is checked

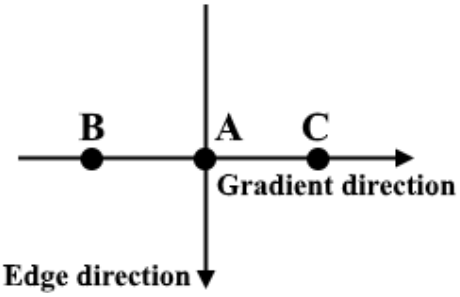

Figure 17 Non-maximum suppression on three points. Reprinted from *Towards Data Science* website, by C. Lin, 2018, retrieved from https://towardsdatascience.com/tutorial-build-a-lane-detector-679fd8953132

whether it is the local maximum in its neighborhood, in the direction of the gradient (Lin, 2018) (see Figure 17). To check if point A is a local maximum, it is compared with points B and C because B and C are in the gradient direction. If point A is a local maximum it is considered in the next stage, if not it, the pixel value of A is set to zero and A is suppressed.

(4) Hysteresis thresholding. To decide which edges are really edges, two pre-defined threshold values are used. These two thresholds are minVal and maxVal. Every pixel with an intensity gradient higher than the maxVal is classified as an edge and every pixel with an intensity gradient lower than the minVal is classified as a non-edge. The pixels with an intensity gradient between minVal and maxVal are classified based on their connectivity. These pixels will be classified as an edge if they are connected with pixels with an intensity gradient above the maxVal. See Figure 18, point A is classified as edge because it is above the maxVal. Point C, which is between the minVal and maxVal, will also be classified as an edge, because of its connection with point A. Point B, which is between the minVal and maxVal will be classified as a non-edge because it has no connection with a point above the maxVal (Lin, 2018). To detect the edges, it is therefore important to set the minVal and maxVal properly (OpenCV, 2015).

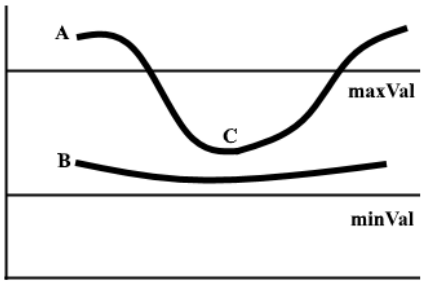

Figure 18 Hysteresis thresholding example on two lines. Reprinted from *Towards Data Science* website, by C. Lin, 2018, retrieved from https://towardsdatascience.com/tutorial-build-a-lane-detector-679fd8953132

In this thesis, multiple images are processed. All these images differ from each other concerning content and they are captured under changing lighting circumstances. Therefore, the optimal values for minVal and maxVal differ per image (Rosebrock, 2015). To overcome this challenge the parameters minVal and maxVal were set per image. First, the median of the image was computed. MinVal was set by multiplying the median with 1 – sigma. MaxVal was set by multiplying the median with 1 + sigma. Where sigma was 0.33. This value of sigma managed to get good results (Rosebrock, 2015).

3.4.1.2 Hough transform

A straight line can be expressed by two parameters. The most widely used and simplest is parameter pair (a, b). In the Cartesian system, a corresponds to the slope and b to the intercept. A line can be described as $y = ax + b$. A line can also be expressed in the polar system with parameter pair (ρ, θ). Parameter ρ is the shortest distance between the origin and the line. Parameter θ is the angle between this distance line and the x-axis (Figure 19) (Kacmajor, 2017). A benefit of representation in the polar system over the Cartesian system is that with parameters (ρ, θ) a vertical line can be described, which is impossible with the parameters of the Cartesian system. For Hough transforms, lines are expressed in the polar system. For a given line the specific ρ and θ can be determined. A line equation in the polar system for point x_i, y_i on the given line can be described as: $\rho = x\cos(\theta) + y\sin(\theta)$ (Kacmajor, 2017).

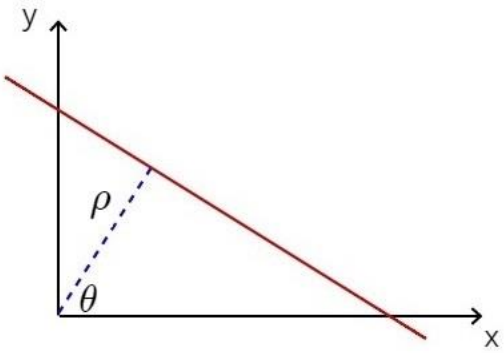

Figure 19 [Representation of a line in the polar system]. Reprinted from *Progg Blogg* website, by T. Kacmajor, 2017, retrieved from https://tomaszkacmajor.pl/index.php/2017/06/05/hough-lines-transform-explained/

A line in the image space, which is represented by ρ and θ, can be drawn as a point (ρ, θ) in the Hough space (Figure 19). If you draw an infinite number of other lines in the image space, which all go to one common point, the corresponding points from these lines, form a continuous sinusoid in the Hough space (Figure 20). Following this, it can be said that a straight line in the image space is a point in the Hough space and a point in the image space is a sinusoid in the Hough space (Kacmajor, 2017).

If you draw points in the image space, which form a line, you obtain several sinusoids in the Hough space, which all intersect at one point (Figure 21). Therefore, to identify a straight line you should look at the intersections in the Hough space (Kacmajor, 2017). The more sinusoids intersect at a point, the more points the straight line has. A threshold can be defined to decide the minimum number of intersections that are needed to detect a line (Bradski & Kaehler, 2008). Thus, Hough transform maintains the intersections between the sinusoids of every point in the image. If the number of intersections succeeds the threshold, Hough transform declares it as a line, with parameters (ρ, θ) of the intersection point.

The parameter setting of the standard Hough line transform in this thesis was as follows. The accuracies of ρ and θ were set at their default. For the parameter threshold, different values were tested. These values were evaluated by comparing five different frames, in different settings. The best results were with a threshold of 175. The Hough line transform was evaluated by manually reviewing the results of 100 random images and determine whether the lines in the image were detected or not. The evaluation metrics used are precision, recall, and F1-score. Accuracy is not used as a metric because we do not

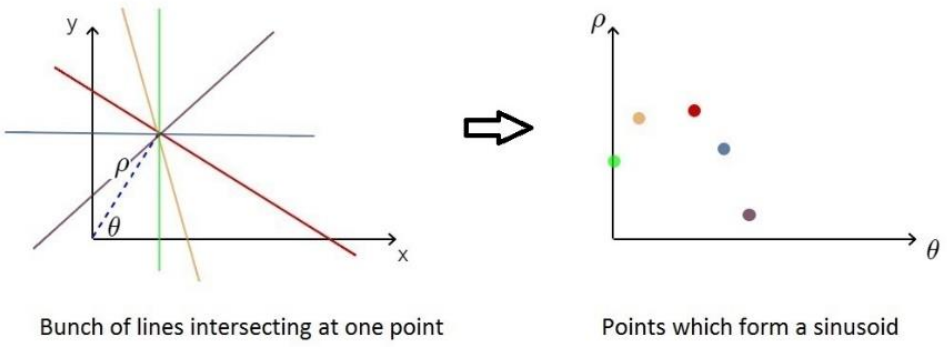

Figure 20 [Infinitive number of lines through one point in image space form sinusoid in Hough space]. Reprinted from *Progg Blogg* website, by T. Kacmajor, 2017, retrieved from https://tomaszkacmajor.pl/index.php/2017/06/05/hough-lines-transform-explained/

determine true negatives as every pixel that is not a line and is not detected as a line would be considered as true negative.

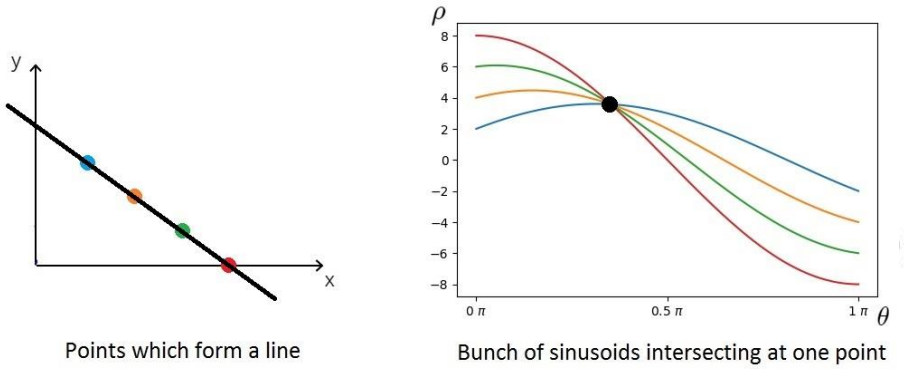

Figure 21 [Points from line in image space form sinusoids in Hough space]. Reprinted from *Progg Blogg* website, by T. Kacmajor, 2017, retrieved from https://tomaszkacmajor.pl/index.php/2017/06/05/hough-lines-transform-explained/

3.4.2 Progressive Probabilistic Hough transform

The standard Hough transform is a robust statistical method to extract lines, but it costs a lot of computation to iterate over all the points and vote whether it is a line or not (OpenCV, 2014). The optimization of the Hough transform to reduce computation is the PPHT. PPHT has the same pipeline as the standard Hough line transform (Figure 16). The difference is that PPHT does not examine all the points, but randomly selects a subset of the points. It has been proven that this smaller set of points is sufficient for line detection (Matas et al., 2000). PPHT, therefore, decreases computation time, while retaining accuracy. The implementation of PPHT in OpenCV is based on the work of Matas et al. (2000).

One of the drawbacks of PPHT is that it fails to find short line segments. This should not be a problem within this thesis, because the goal is to find shelves and cupboards, which are long line segments. The Hough line transform was evaluated by manually reviewing the results of 100 random images and determine whether the lines in the image were detected or not. The evaluation metrics used are precision, recall, and F1-score.

4. Results

4.1 Annotation

We have annotated the cups in the dataset with interactive tracking. Sometimes the tracker losses the cup. The main reason for losing the cup was that the cup was not in the frame anymore. Twenty-two times the tracker lost the cup because the bounding box was slowly moving away or because the camera was moving too fast. Every time the tracker lost the cup the bounding box was adjusted manually. The coordinates of the bounding boxes were used to crop the cups from the images and a database of cups was created. The results of the annotation of the cups are shown in the video, which is specified in section 4.4 of this report.

4.2 Object Detection

In this section, the result of the object detectors will be presented. First, we present the results of template matching (Table 1). The frames used for template matching were not randomly selected, because it was necessary for a cup to be present in all the frames. To ensure that there was a variation between the frames, frames were selected that differ from each other in lightning condition, objects visible, and the number of cups present. The precision is 0,225, the recall is 1,000 and the F1-score is 0.368. The recall is high because all the templates were cups, so there were no false negatives. In most of the frames, the algorithm was unable to detect the cup in the image. In Figure 22 the output of a bad template matching, with the corresponding template, is shown. In Figure 23 the output of a good template matching, with the corresponding template, is shown.

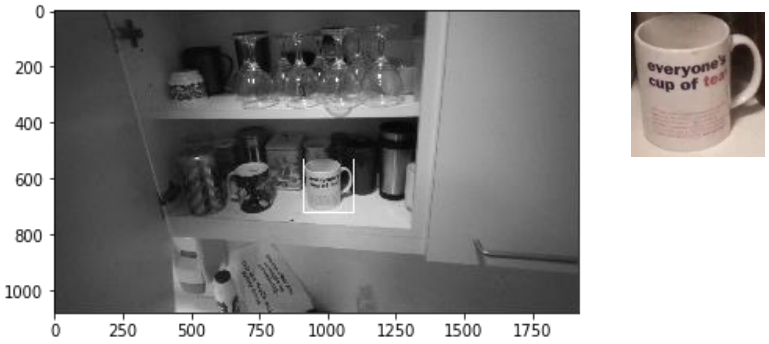

Figure 22 Bad template matching result

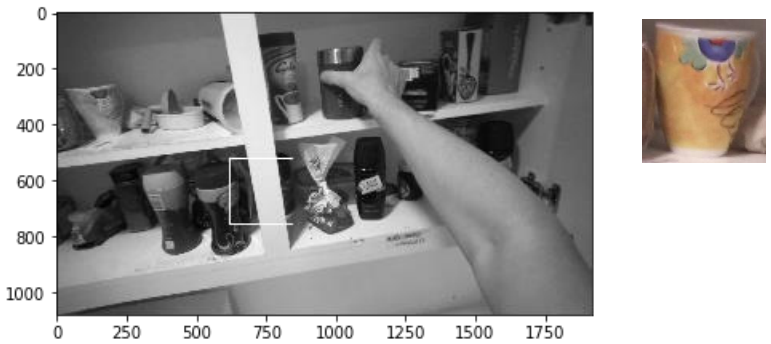

Figure 23 Good template matching result

Second, the results of RetinaNet are presented. In Table 1 the results of RetinaNet on 50 random frames are presented. The minimal probability was 50% and the only object that was modeled to detect was a cup. In Figure 24 two output frames are shown. In the left frame, the model detected the cup in the image. In the right frame, RetinaNet detected one cup in the image but did not detect the cup with the butterflies in the image. The recall of cup detection is 0.367, the precision is 0.688, and the F1-score is 0.479.

Table 1 Confusion matrix template matching and RetinaNet

Method	Confusion matrix			Evaluation parameter		
Template matching		Correct labels		Precision	Recall	F1-score
		Positive	Negative			
	Positive	72	248	0.225	1.00	0.368
	Negative	0	NaN			
RetinaNet (cups)		Correct labels		Precision	Recall	F1-score
		Positive	Negative			
	Positive	11	5	0.688	0.367	0.479
	Negative	19	18			
RetinaNet (eleven objects)		Correct labels		Precision	Recall	F1-score
		Positive	Negative			
	Positive	17	8	0.680	0.327	0.442
	Negative	35	143			

Figure 24 RetinaNet object detection cups

Furthermore, RetinaNet was modeled to detect eleven custom objects with a minimal probability of 50%. In Appendix A the confusion matrix with all the eleven objects on 100 randomly selected frames are shown. An example output frame is presented in Figure 25. The objects to detect were: cup, person, fork, knife, spoon, bottle, bowl, chair, table, toaster, sink, and refrigerator. The detection of cups, while considering other objects, gives a precision of 0.680, a recall of 0.327, and an F1-score of 0.442.

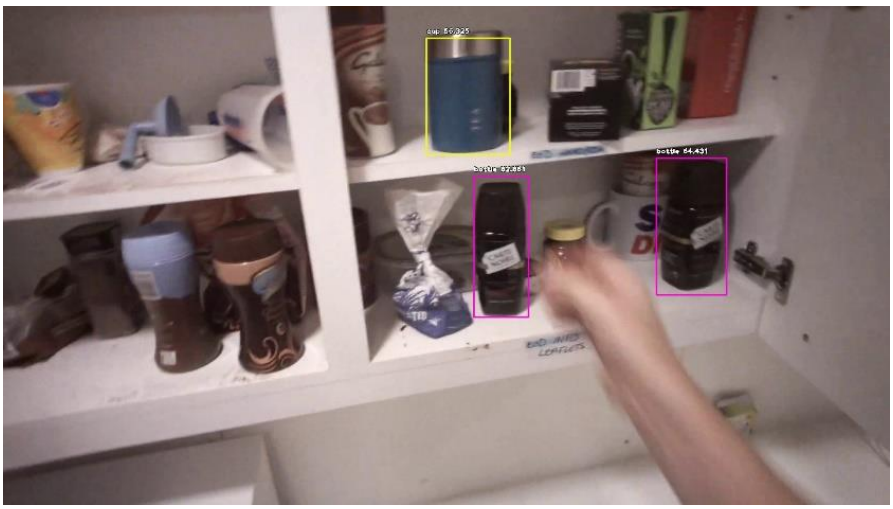

Figure 25 RetinaNet object detection on multiple custom objects. Yellow box = cup, pink box = bottle.

4.3 Line Detection

In this section, the results of the standard Hough line transform and the PPHT will be described. The task of both the Hough line transforms was to find the lines in the image. These lines represented the shelves and cupboards in the images. Hundred random frames were evaluated to construct the confusion matrixes shown in Table 2. First, the results of the standard Hough line transform will be discussed. The performance of the standard Hough line transform in detecting long straight lines was bad, with a recall of 0.060, precision of 1, and an F1-score of 0.113. In 94 frames the algorithm did not find all the lines in the image. Only in six frames, the line detector was able to find all the lines present. In these frames, there was only one clear, long line present (see Figure 26).

Figure 26 Standard Hough line transform

Table 2 Confusion matrix standard Hough transform and PPHT

Method	Confusion matrix			Evaluation parameter		
Standard Hough transform		Correct labels		Precision	Recall	F1-score
		Positive	Negative			
	Positive	6	0	1.000	0.060	0.113
	Negative	94	NaN			
PPHT		Correct labels		Precision	Recall	F1-score
		Positive	Negative			
	Positive	46	49	0.484	0.939	0.639
	Negative	3	NaN			

For PPHT, the parameters minLineLenght and maxLineGap, were tested with different values. These values were evaluated by comparing five different frames, in different settings. The parameters with the best results were a minLineLenght of 175 and a maxLineGap of 300. The performance of the PPHT technique was not always optimal, with an F1-score of 0.639, a recall of 0.939, and a precision of 0.484 (Table 2). In 49 frames, where we as a human could detect the lines, the algorithm did not find any lines or just found some lines, but not all the lines that are present in the image. Some examples of these frames are presented in Figure 27. In 46 frames the algorithm was able to detect all the long lines present in the image. Some example frames where the PPHT performed well are shown in Figure 28.

Figure 27 PPHT with no or a few lines detected

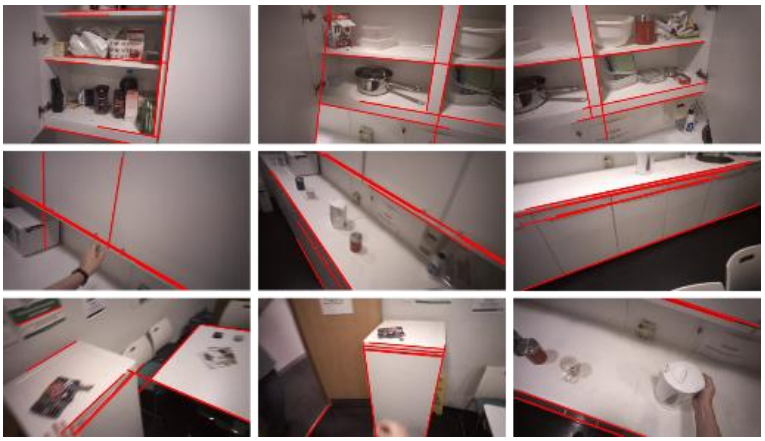

Figure 28 PPHT good performance

From the hundred random images, in three images the algorithm found more lines than there were. These three images are shown in Figure 29. In two frames the PPHT algorithm detected too many as well as too few lines. Figure 30 shows one of the two frames. In this frame, you see that there are some lines detected in places where there is no line, like on the ground. But it also did not detect lines that are present in the image, like the lines of the fridge. These two outcomes are not included in the confusion matrix in Table 2, because they cannot be categorized in one of the cells, because they are false positive as well as false negative.

Figure 29 PPHT to much lines detected.

Figure 30 PPHT of too many and too little lines detected.

To see whether the detection of too many lines was due to the wrong setting of the minVal and maxVal, three successive frames were examined. Successive frames would have almost the same minVal and maxVal because there are captured under comparing lighting conditions and the frames would contain almost the same kind of objects. In Figure 31, three successive frames are shown. It was expected that they would show identical Hough lines, but this is not the case.

Another reason for the detection of too many lines could be that the problem lies earlier in the pipeline, with the Canny edge detection. Therefore, the corresponding outputs of the edge detection of the three successive frames were examined. The output is shown

in Figure 31. From this output, it cannot be said that the problem of finding too many lines is due to edge detection because the Canny edge detection output looks the same for the three images.

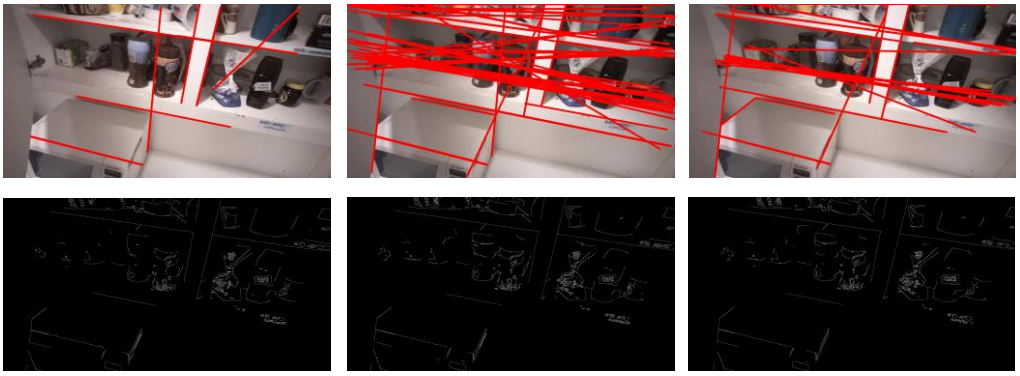

Figure 31 Three successive frames with PPHT and Canny edge detection.

4.4 Video of PPHT and tracking of cups

To combine the two research questions and to show the results of the PPHT for all frames a video was made. We applied PPHT to all frames to find the shelves and cupboards. For all the detected lines a red line was drawn over the input frame and the outcomes were put in a video. To show how the cups in the video relate to the detected shelves, a green bounding box was drawn around the cups. These bounding boxes were tracked over time and adjusted if needed, with the use of OpenCV Tracker. The video can be found at https://youtu.be/gxMZkBXi6Sk and a screenshot of the video is shown in Figure 32. The results in the video correspond with the results of the PPHT on the 100 randomly selected frames, in the degree of accuracy.

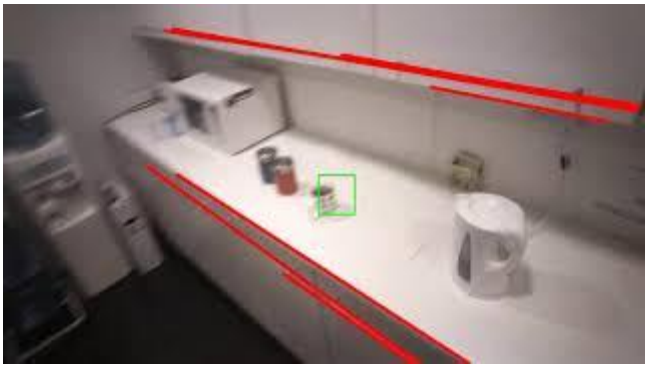

Figure 32 Screenshot from video with detected lines and cups

5. Discussion

The goal of this study was to examine to what extent RetinaNet has improved cup detection in a tea making scene and to examine which line detection method performs best at detecting the line boundaries of pantry furniture in a tea making scene. Furthermore, this work studies how well tracking approaches improve the annotation of cups in a video of tea making.

First, the results of the annotation of the cups were presented. The goal was to answer the sub research question: Could tracking approaches help the manual annotation of cups in our dataset? For this study, the answer is yes, they can. Tracking reduces the annotation time substantially while retaining good accuracy. A limitation of manually correcting a drifting tracker is that you must watch the entire video (Vondrick & Ramanan, 2011). In this thesis, this is not a problem because the analyzed video was relatively short, but this should be kept in mind when choosing an annotation method. The annotation was performed to create a database of cups, which was used for template matching. Another option was to use an existing database with cups. The reason for creating an own database was that template matching considers all pixels in the template, so also the background in the template. The templates in the self-created database have more resemblance to the input images than an existing database. Furthermore, the use of an existing database requires a lot of storage space. So, the expectation was that the use of a self-created database would lead to better results. The performance of template matching with an existing database was not analyzed. A recommendation is to analyze this in future research.

Second, the results of the object detectors were presented. The research question about object detection was: To what extent does RetinaNet improves cup detection in video recordings acquired from a participant's point of view, while they are making tea? The baseline classifier template matching had an F1-score of 0.368. However, the reason that this score is not lower is due to a recall of 1.00. The recall was 1.00 because there were no false negatives with template matching. Since all templates were of cups. So, it makes more sense to compare the precision of template matching and RetinaNet. The precision of template matching was 0.225. A low precision was expected because template matching only works if the object to detect is very simple and does not change much in appearance (Mallick, 2017). This is not the case with the data of this thesis. There are multiple cups in the video, sometimes even multiple cups in one frame and the appearance of the cup changes through the video. On the other hand, RetinaNet had a precision of 0.688. This is a substantially higher precision than the baseline. This is in accordance with related work, that found that neural networks are the best object detectors methods (Janai, et al., 2017). We have applied RetinaNet for cup detection, but also for detections of eleven different objects. When detecting eleven different objects RetinaNet had a higher accuracy (0.788) than when it was only detecting cups (0.547). The higher accuracy of the eleven different objects is mostly due to the contribution from true negatives. A reason for this could be that the model RetinaNet is better trained at detecting other objects than cups or that the cups that the model is trained on differ from the cups in the video. This could mean that RetinaNet may not be the most suitable for object detection in a tea making video. To see if this is the case RetinaNet can be compared with other neural networks, in future work, to examine if the difference in accuracy is due to the model or to the data used. This thesis contributes to the scientific literature by specifically look to cup detection in videos from the first-person view.

Third, the results of the standard Hough line transform and the PPHT were presented to answer the research question: How well does PPHT performs compared to the standard Hough transform at detecting line boundaries of pantry furniture in the same video recordings? The standard Hough line transform had a very low F1-score of 0.113 while the PPHT had an F1- score of 0.639. Therefore, PPHT is a better method for detecting lines in

a tea making scene. This is in accordance with related work that found that PPHT performs better at line detection than the standard Hough transform (Von Gioi et al., 2010; Matas et al., 2000). The PPHT resulted in a high recall of 0.939. However, it had a lower precision of 0.484. In most frames, the PPHT was unable to detect all the lines in the image. In most cases when it was unable to detect the lines, the input image was a moving image and not a sharp image. There were a lot of blurry frames because the participant was making tea, and this involved a lot of moving around to find objects and therefore, moving the camera. The presence of blurry images could be the reason for the low precision. Furthermore, from the results, it could not be said whether the detection of too many lines was due to edge detection or another reason. It was assumed that successive frames share the same minVal and maxVal parameter. But maybe this is not the case in these frames. This is a limitation of this study. A recommendation for future research is to examine what the reason is for the detection of too many lines in an image. This thesis contributes to the current work by specifically look to line detection of pantry furniture.

Furthermore, a limitation of this research is that only the video of one participant was analyzed. This has a negative influence on the generalizability of the study. Only one video was analyzed due to time reasons. In future research, more videos should be analyzed to see whether the results are applicable to other videos or whether there are other reasons for these results.

Another limitation is the parameter setting. The parameter values of template matching, standard Hough transform and PPHT were chosen based on the results of only a couple of frames. The frames for parameter setting were carefully chosen in a way that they included different settings, lighting conditions, and the number of cups and lines present. However, it could be that these frames weren't the best representation of the whole video and that different parameter settings would have led to different results.

6. Conclusion

The research questions of this thesis were: To what extent does RetinaNet improves cup detection in video recordings acquired from a participant's point of view, while they are making tea? And: How well does PPHT performs compared to the standard Hough transform at detecting line boundaries of pantry furniture in the same video recordings? To answer these questions different object detection and line detection methods were applied to video data, of 3401 frames, where the task was to make tea. The answer to the first research questions is that state-of-the-art object detector RetinaNet performs substantially better than the baseline classifier, template matching, in the task of cup detection. And the answer to the second research question is that the PPHT has a better performance at detecting the line boundaries of pantry furniture, than the standard Hough transform. These results are both in accordance with existing research. The sub research question of this thesis was: Could tracking approaches help the manual annotation of cups in our dataset? The interactive tracking approach was substantially faster at video annotation than manual annotation while retaining good accuracy. Future research could focus on improving line detection, possible with the use of tracking. Another recommendation for future research is to concentrate on detecting more and other objects than cups and lines as well as semantic segmentation methods to detect pantry furniture. Furthermore, future research could focus on the further development of the combinations of the detected lines and cups to be able to tell where the cups are in the environment. Since the data is gathered with an eye-tracker future research could combine the video data with gaze points to see where the participants are looking at.

References

Adhikari, B., Peltomaki, J., Puura, J., & Huttunen, H. (2018). Faster bounding box annotation for object detection in indoor scenes. In *2018 7th European Workshop on Visual Information Processing (EUVIP)* (pp. 1-6). IEEE.

Agarwala, A., Hertzmann, A., Salesin, D. H., & Seitz, S. M. (2004). Keyframe based tracking for rotoscoping and animation. In *ACM Transactions on Graphics (ToG)* (Vol. 23, No. 3, pp. 584-591). ACM.

AI Stanford. (n.d.). *Introduction to Computer Vision.* Retrieved from http://ai.stanford.edu/~syyeung/cvweb/tutorial1.html

Ali, S., Al Mamun, S., Fukuda, H., Lam, A., Kobayashi, Y., & Kuno, Y. (2018). Smart Robotic Wheelchair for Bus Boarding Using CNN Combined with Hough Transforms. In *International Conference on Intelligent Computing* (pp. 163-172). Springer, Cham.

Barrow, H.G., & Tenenbaum, J.M. (1981). Interpreting line drawings as three-dimensional surfaces. *Artificial Intelligenge, 17*(1-3), 75-116

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.".

Brunelli, R. (2009). *Template matching techniques in computer vision: theory and practice*. John Wiley & Sons.

Carrasco M. (2011).Visual attention: The past 25 years. *Vision Research, 51*(13), 1484-1525. doi:10.1016/j.visres.2011.04.012

Cartucho, J. (2018). *OpenLabeling: open-source image and video labeler.* Retrieved from https://github.com/Cartucho/OpenLabeling

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems* (379-387).

Dia Scan. (2013). *Digitized picture correction with digiKam – Part 2*. Retrieved from http://dia-scan.blogspot.com/2013/02/digitized-picture-correction-with-digiKam-part2.html

Elboher, E., & Werman, M. (2013). Asymmetric correlation: a noise robust similarity measure for template matching. *IEEE Transactions on Image Processing*, *22*(8), 3062-3073.

Elsalamony, H. A. (2015). Detecting distorted and benign blood cells using the hough transform based on neural networks and decision trees. In *Emerging Trends in Image Processing, Computer Vision and Pattern Recognition* (457-473). Morgan Kaufmann.

George, M., Jose, B. R., & Mathew, J. (2018). Performance Evaluation of KCF based Trackers using VOT Dataset. *Procedia Computer Science*, *125*, 560-567.

Girshick, R. (2015). Fast R-CNN ICCV. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)* (1440-1448).

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Graetz, F.M. (2018). *Retinanet: how focal loss fixes single-shot detection.* Retrieved from https://towardsdatascience.com/retinanet-how-focal-loss-fixes-single-shot-detection-cb320e3bb0de

Gumgum, (2016). *How computer vision works (and why it's so amazing).* Retrieved from https://gumgum.com/what-is-computer-vision

Hashemi, N. S., Aghdam, R. B., Ghiasi, A. S. B., & Fatemi, P. (2016). Template Matching Advances and Applications in Image Analysis. *arXiv preprint arXiv:1610.07231*.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the*

IEEE international conference on computer vision (2961-2969).
He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (770-778).
Hernández, A., Gómez, C., Crespo, J., & Barber, R. (2016). Object detection applied to indoor environments for mobile robot navigation. *Sensors*, *16*(8), 1180.
Humans in the loop. (2018). The best image annotation tool platforms for computer vision (+ an honest review of each). Retrieved from https://hackernoon.com/the-best-image-annotation-platforms-for-computer-vision-an-honest-review-of-each-dac7f565fea
Huynh, K. (2017). *What does "class-agnostic" in most of the object detection papers mean?*. Retrieved from https://www.quora.com/What-does-%E2%80%9Cclass agnostic%E2%80%9D-in-most-of-the-object-detection-papers-mean
Idrees, H., Shah, M., & Surette, R. (2018). Enhancing camera surveillance using computer vision: a research note. *Policing: An International Journal*, *41*(2), 292-307.
Ioannidou, F., Hermens, F., & Hodgson, T. (2016). The centrial bias in day-to-day viewing. *Journal of Eye Movement Research*, *9*(6), 1-13.
Jain, A. (2016). *Deep learning for computer vision – Introduction to convolutional neural networks.* Retrieved from https://www.analyticsvidhya.com/blog/2016/04/deep learning-computer-vision-introduction-convolution-neural-networks/
Janai, J., Güney, F., Behl, A., & Geiger, A. (2017). Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. *arXiv preprint arXiv:1704 .05519*.
Jay, P. (2018). *The intuition behind retinanet.* Retrieved from https://medium.com/ @14prakash/the-intuition-behind-retinanet-eb636755607d
Kacmajor, T. (2017). *Hough lines transform explained.* Retrieved from https:/ /tomaszkacmajor.pl/index.php/2017/06/05/hough-lines-transform-explained/
Kaehler, A., & Bradski, G. (2016). *Learning OpenCV 3: computer vision in C++ with the OpenCV library*. " O'Reilly Media, Inc.".
Land, M. F., Mennie, N., & Rusted, J. (1999). Eye movements and the roles of vision in activities of daily living: making a cup of tea. *Perception*, *28*(4), 1311– 1328.
Li, F.F., Johnson, J., & Yeung, S. (2017). *Lecture 11: Detection and segmentation* [Slides]. Retrieved from https://github.com/khanhnamle1994/computer-vision/blob/master /Lecture-11-Detection-and-Segmentation/cs231n_2017_lecture11.pdf
Li, Y., Chen, L., Huang, H., Li, X., Xu, W., Zheng, L., & Huang, J. (2016). Nighttime lane markings recognition based on Canny detection and Hough transform. In *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)* (411-415). IEEE.
Lin, C. (2018). *Tutorial: Build a lane detector*. Retrieved from https://towardsdatascience .com/tutorial-build-a-lane-detector-679fd8953132
Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2117-2125.
Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2980-2988.
Long, X. (2018). *Understanding object detection in deep learning.* Retrieved from https: //blogs.sas.com/content/subconsciousmusings/2018/11/19/understanding-object-detection-in-deep-learning/
Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (3431-3440).

Maladkar, K. (2018). *Computer vision primer: How ai sees an image.* Retrieved from https://www.analyticsindiamag.com/computer-vision-primer-how-ai-sees-an-image/

Mallick, S. (2017). *Object Tracking using opencv (C++/Python).* Retrieved from https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/

Matas, J., Galambos, C., & Kittler, J. (2000). Robust detection of lines using the progressive probabilistic hough transform. *Computer Vision and Image Understanding*, *78*(1), 119-137.

Mihailidis, A., Carmichael, B., & Boger, J. (2004). The use of computer vision in an intelligent environment to support aging-in-place, safety, and independence in the home. *IEEE Transactions on information technology in biomedicine*, *8*(3), 238-247.

Olafenwa, J., & Olafenwa, M. (2018a). *Imageai.* Retrieved from https://github.com/OlafenwaMoses/ImageAI

Olafenwa, J., & Olafenwa, M. (2018b). *Introduction to deep computer vision*. Retrieved from https://john.aicommons.science/deepvision/

OpenCV. (2013). *Template matching*. Retrieved from https://opencv-pythontutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_template_matching/py_template_matching.html

OpenCV. (2014). *Hough line transform*. Retrieved from https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html

OpenCV. (2015). *Canny edge detection.* Retrieved from https://docs.opencv.org/3.1.0/da/d22/tutorial_py_canny.html

OpenCV. (2018). *OpenCV modules.* Retrieved from https://docs.opencv.org/4.0.0/index.html

Oron, S., Dekel, T., Xue, T., Freeman, W. T., & Avidan, S. (2018). Best-buddies similarity—robust template matching using mutual nearest neighbors. *IEEE transactions on pattern analysis and machine intelligence*, *40*(8), 1799-1813.

Perveen, N., Kumar, D., & Bhardwaj, I. (2013). An overview on template matching methodologies and its applications. *International Journal of Research in Computer and Communication Technology*, *2*(10), 988-995.

Photoworld, (2015). *How big is snapchat?* Retrieved from https://cewe-photoworld.com/how-big-is-snapchat/

Ren, S., He, K., Girshick, R. B., & Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*

Rosebrock, A., (2015). *Zero-parameter, automatic canny edge detection with python and opencv.* Retrieved from https://www.pyimagesearch.com/2015/04/06/zeroparameter-automatic-canny-edge-detection-with-python-and-opencv/

Seifozzakerini, S., Yau, W. Y., & Mao, K. (2017). Effect of inhibitory window on event-based Hough transform for multiple lines detection. In *Proceedings of the International Conference on Advances in Image Processing* (39-44). ACM.

Sermanet, P., Kavukcuoglu, K., Chintala, S., & LeCun, Y. (2013). Pedestrian detection with unsupervised multi-stage feature learning. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Shi, W., & Samarabandu, J. (2006). Corridor line detection for vision based indoor robot navigation. In *2006 Canadian Conference on Electrical and Computer Engineering*(1988-1991). IEEE.

Suárez, I., Muñoz, E., Buenaposada, J. M., & Baumela, L. (2018). FSG: A statistical approach to line detection via fast segments grouping. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (97-102). IEEE.

Vondrick, C., & Ramanan, D. (2011). Video annotation and tracking with active learning. In *Advances in Neural Information Processing Systems* (28-36).

Vondrick, C., Ramanan, D., & Patterson, D. (2010). Efficiently scaling up video annotation with crowdsourced marketplaces. In *European Conference on Computer Vision* (610-623). Springer, Berlin, Heidelberg.

Von Gioi, R. G., Jakubowicz, J., Morel, J. M., & Randall, G. (2010). LSD: A fast line segment detector with a false detection control. *IEEE transactions on pattern analysis and machine intelligence*, *32*(4), 722-732.

Yuen, J., Russell, B., Liu, C., & Torralba, A. (2009). Labelme video: Building a video database with human annotations. In *2009 IEEE 12th International Conference on Computer Vision* (1451-1458). IEEE.

Zeng, N. (2018). *Retinanet explained and demystified*. Retrieved from https://blog.zenggyu.com/en/post/2018-12-05/retinanet-explained-and-demystified/

Zheng, F., Luo, S., Song, K., Yan, C. W., & Wang, M. C. (2018). Improved Lane Line Detection Algorithm Based on Hough Transform. *Pattern Recognition and Image Analysis*, *28*(2), 254-260.

Appendix A: Confusion matrix RetinaNet, 50% minimal probability, 100 random frames, 11 custom objects.

Actual	Cup	Person	Fork	Knife	Spoon	Bottle	Bowl	Chair	Toaster	Sink	Refrigerator	Another object
P Cup	17											8
P Person		31										2
P Fork												1
P Knife												
P Spoon												2
P Bottle						5						8
P Bowl	1						9					3
P Chair												1
P Toaster												
P Sink										5	1	7
P Refrigerator											2	2
P Nothing	36	8			10	16	1	7		14	1	