WHAT AM I LOOKING AT?

TOWARDS AUTOMATIC VIDEO ANNOTATION FOR MOBILE EYE TRACKING RESEARCH

Master's Thesis (27,5 ECTS)
Applied Cognitive Psychology
Utrecht University

Rogier Simons
5531179
July 2021

First Examiner: Ignace T.C. Hooge
Second Examiner: Jeroen S. Benjamins

Abstract

The developments of wearable eye trackers in Eye Tracking research allowed for more freedom in the experimental setup, but at the cost of extensive manual analysis time. This research tried to decrease manual analysis time of mobile eye tracking experiments by developing a technique for semi-automatic annotation of the video material. First, different approaches proposed by other research were analysed, to find possible improvements in the current solutions. Using object detection models seemed most promising, but a big problem with this technique are the extensive training sets that are required to train these models. This research proposes a new way to annotate a small part of the video material, which can be used to create the required training sets with less manual effort than in traditional annotation tools. In a preliminary comparison to the completely manual annotation process an object detection model was trained that was was able to label around 70% of the Areas of Interest in the video material. In this comparison the proposed approach became feasible after multiple participants, when the initial time it takes to gather the training data weighs up to the manual annotation of the gaze fixations, but from that moment this approach seems quicker than other proposed solutions for this problem, with the added freedom of training on custom objects. The new approach is still in its early days, but the proposed combination of techniques seem to be quite promising.

Keywords – Mobile Eye Tracking, Eye Tracking, Automatic Video Annotation, Object Tracking, Object Detection

Contents

Ut imago est animi voltus sic indices oculi -
The face is a picture of the mind as the eyes are its interpreter.

Cicero (106-43 B.C.)

1 Introduction

1.1 Motivation

Eye Tracking has been a valuable research tool for many years in a large variety of areas (Jacob and Karn, 2003; Karthikeyan et al., 2013; Majaranta and Bulling, 2014; Rayner, 1998; Sharafi et al., 2020). With the ability to measure gaze patterns of human participants, it enables analysis of behavioural patterns that take place on a subconscious level (Asman et al., 2019). Developments in eye tracking technology opened up a broad range of new possibilities in this research area, but also introduced a new set of problems. This section will introduce the scope of eye tracking research, and explain the research problem this thesis will focus on.

1.1.1 Why Tracking? - Eye Tracking Applications

The human eye can be used to indicate direction of attention (Posner, 1980). Therefore, conducting eye tracking experiments can be used to estimate how people interact with their external environment, or to get insights on their mental well-being. Some real life applications for eye tracking experiments are clinical diagnosis of patients (Harezlak and Kasprowski, 2018; Larrazabal et al., 2019), usability research (Bojko, 2013, p. 24) and research on reading behaviour (Jarodzka and Brand-Gruwel, 2017).

A great overview of the different types of psychological research that can be conducted based on the eye tracking technology was made by Mele and Federici (2012). They focus on five different areas: visual search, reading, natural tasks, scene viewing and other information processing, and give some examples for each area. With such a broad range of possible applications, different scenario's require different tools to carry out the experiments.

1.1.2 Different types of Eye Trackers

There are multiple types of eye trackers, each optimised for different situations (Holmqvist et al., 2011, p. 51-53). The three most popular options are the tower-mounted tracker, the remote eye tracker and the head-mounted tracker (Figure 1). Holmqvist mentions the different types of eye tracker come with their own advantages and disadvantages in freedom, data quality and data analysis (Table 1). The tower-mounted tracker consists of a chin- and forehead rest to keep the participant stationary relative to a (infra-red) camera and a screen. The advantage of this set up is the high quality of the recording of the eye movements, enabled by the limited movement by the participant relative to the camera. There are situations where this limited movement is undesirable or even impossible, for example with infant research, where this limited movement can be considered a downside. The remote eye tracker does not require this chin- and forehead rest. Advantages of this set up are that it requires less instruments from the researcher making it more accessible to use, and that the participants have a little more freedom

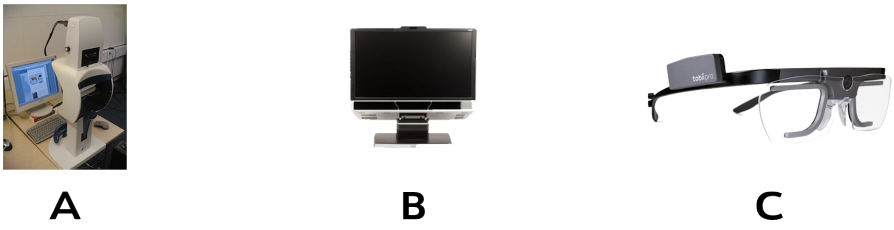

Figure 1: Different types of eye trackers: A) Tower Mount, B) Remote, C) Head Mount

to move. A downside is that this freedom decreases the quality of the eye movement recordings. The head-mounted eye tracker will be discussed in the next section.

	Freedom	Data Quality	Data Analysis
Tower Mount	--	++	++
Remote	+	-	++
Head Mount	++	--	--

Table 1: (Dis)Advantages of different eye trackers

In traditional Eye Tracking experiments participants are positioned in front of a fixed camera that is filming the eyes of the participants while they look at some visual material (tower-mounted and remote eye trackers). This setup allows no to little movement of the location of the participants. Developments in the technology that supports the eye tracking experiments allow for more freedom in the experimental set up (Holmqvist et al., 2011, p. 51), with eye trackers that are attached to a head mount. This thesis will from here on focus on the head-mounted eye tracker, and specifically on the great disadvantage in its data analysis.

1.1.3 Mobile Eye Tracking

Advancements in Eye Tracking technologies allowed the researcher to leave the fixed experimental setup where the participant had to remain in a fixed position in relation to the recorder. Head-mounted eye trackers (or Mobile Eye Trackers) consist of a front-facing 'scene camera' that films the world in front of the participant. The 'eye camera' is a near-infrared camera (or multiple depending on which model eye tracker is used) in combination with near-infrared lights directed towards the eye and is used to estimate the gaze direction (Figure 2). This setup allowed experiments to take place in environments such as shops, public facilities and working environments (Harwood and Jones, 2014; Hasanzadeh et al., 2016; Weibel et al., 2012). Wearable glasses with cameras recording the participant's

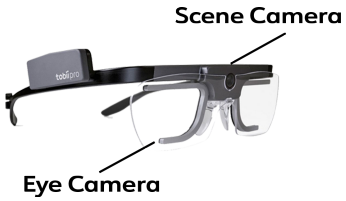

Figure 2: Head-mounted Eye tracking camera's

view and eye movements allow participants to roam freely through these environments, encouraging natural behaviour in comparison to the more controlled lab settings.

1.1.4 Problems with Mobile Eye Tracking

While there are many benefits of this development, the Mobile Eye Tracking technology also knows many limitations. A current limitation in Mobile Eye Tracking research is that its data analysis is a very time consuming task (Browning et al., 2016; Eghbal-Azar and Widlok, 2013). This is at the least caused by a combination of two specific problems.

To illustrate these two problems *three different scenario's* are introduced, where the goal in each scenario is to estimate how much time is spent looking at an object of interest, in these scenario's depicted by the triangle.

Scenario 1: An experiment can be carried out in the traditional fixed setting, as is the case with tower-mounted or remote eye trackers, with static visual material (Figure 3). An example of this static material can be an advertisement poster. During this experiment the participant's gaze directions are recorded while they are looking at the static material. The gaze directions need to be mapped to regions of interest in the poster. The regions of interest (where the triangle is) on the poster need to be defined only once, because the poster will not change during the experiment or between participants. The mapping of gaze directions over the regions of interest is a pretty straight forward task and can be done automatically by a computer program.

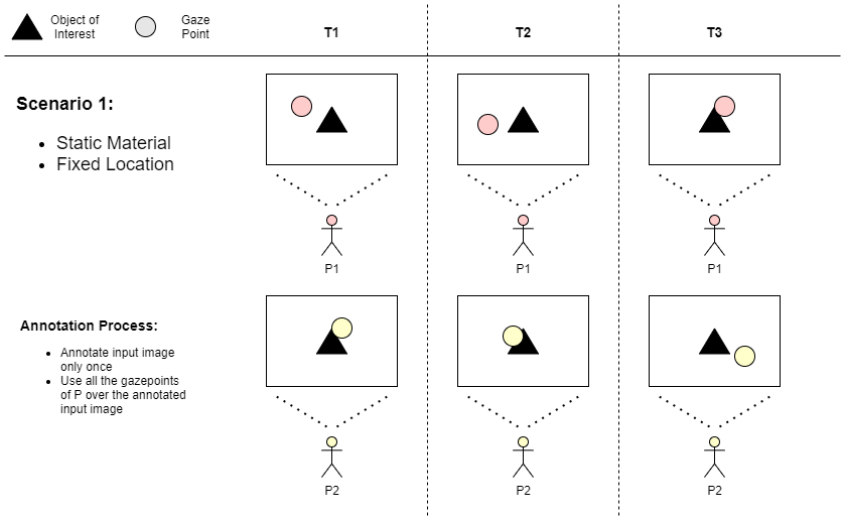

Figure 3: Annotation Process in Scenario 1. This diagram shows the material presented to the participants, where the triangle represents an object of interest. The different columns represent different moments in time during the experiment, and the two rows show the two participants and their different gaze points. As can be seen in this diagram, the position of the object of interest does not change over time nor per participant, so only one annotation of the material is required.

Scenario 2: The first problem arises when the visual material changes over time (Figure 4). The experiment still takes places in the traditional fixed setting, but the visual material may be an advertisement video instead of a poster. Now all the gaze positions can not be mapped to one image like in the first scenario, because the position of the object of interest will change over time. The position of the object of interest will have to be defined in every frame of the video instead of only once like with the poster. After mapping this position in every frame, the combination of gaze directions and regions of interest in each frame can be analysed. The annotation time from one image is multiplied by the amount of frames in the video.

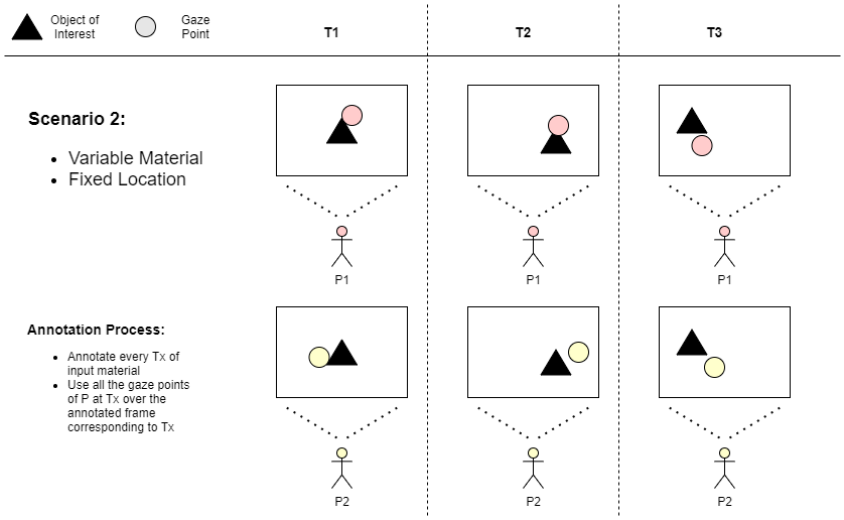

Figure 4: Annotation Process in Scenario 2. As can be seen in this diagram, the position of the object of interest does change over time, but not per participant. All the frames in the material need to be annotated once.

Scenario 3: The second problem arises when the fixed setting is left and the participants are allowed to roam freely in their environment (Figure 5), like with the mobile eye trackers. In the fixed setting, all the frames in the video material would have been annotated once, and that annotation could be used for every participant. In this scenario, the video material (that is captured by the front facing camera) will differ between the participants. Therefore every frame of the video material will have to be annotated, and this has to be repeated for every different participant, meaning the annotation time is multiplied again by the amount of participants.

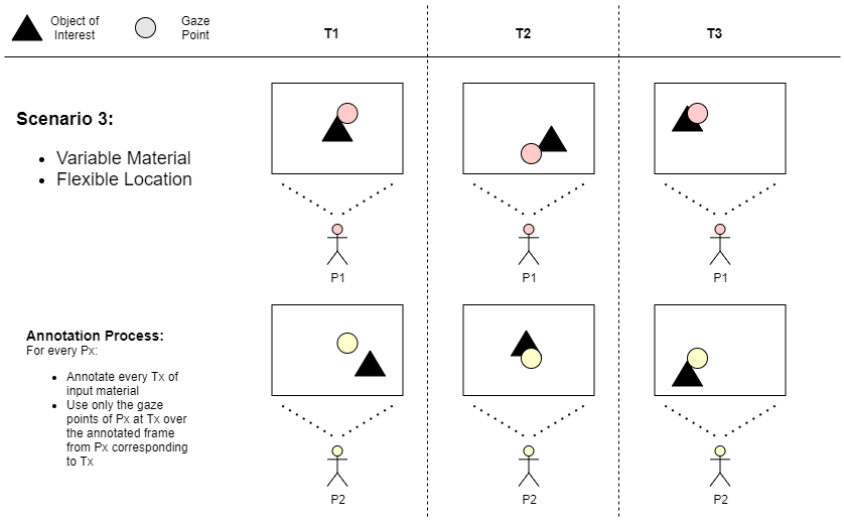

Figure 5: Annotation Process in Scenario 3. As can be seen in this diagram, the position of the object of interest does change over time and per participant, so all the frames in the material need to be annotated and this needs to be repeated for every participant.

Many researchers have tried to tackle the problem of extensive manual labour that is required to conduct these kinds of experiments, with varying rates of success (Batliner et al., 2020; Kurzhals et al., 2016).

1.2 Aim

The aim of this thesis is to decrease analysis time in mobile eye tracking experiments by developing a technique for (semi-)automatic annotation of the video material created by the 'scene' camera's of mobile eye trackers. A decrease in analysis time will allow Mobile Eye Tracking research to be more applicable in many situations, and will relieve the workload of researchers working with mobile eye tracking techniques.

To realise this aim, first existing techniques that are available for analysing the video material will be evaluated. After this assessment, possible improvements on the current techniques will be proposed. The effectiveness of the proposed improved technique will be tested against existing analysis methods of Mobile Eye Tracking experiments. In this way an evaluation of the applicability of the proposed technique will be performed.

2 Related Work

The problem of extensive manual annotation that is required for conducting mobile eye tracking experiments has gotten quite some attention in the last decade, but unfortunately even the state of the art technology is not able yet to independently label video material in the majority of applications. This section will first delve into the solutions other researchers have proposed, to get an understanding of which solutions work, and what directions do not yield promising results. Then, a more in-depth analysis of the technologies that seem promising will be conducted, in order to propose an improved solution for the problem at hand.

2.1 Analysing Eye Tracking Data

In order to draw meaningful conclusions from experiments with eye trackers the gaze direction data from the 'eye camera' has to be combined with the visual material presented to the participants. The gaze directions can be mapped to the visual material to conclude how much time was spent looking at which parts of the material. We call the parts of the material that may be relevant 'Regions of Interest' (ROI) or 'Areas of Interest' (AOI). The latter is more conventional in eye tracking research therefore this terminology will be adopted. As mentioned in the introduction of this thesis, annotating AOI's is still non-trivial for mobile eye tracking research.

2.1.1 Labelling gaze fixations

When performing analysis on mobile eye tracking experiment data, the metrics that are most relevant are what objects were looked at, in which order and for how long (Niehorster et al., 2020). These metrics can be obtained by looking at gaze fixations. There are multiple algorithms for defining gaze fixations, so determining when a fixation is happening is not standardised (Hessels et al., 2017; Larsson et al., 2015; Nyström and Holmqvist, 2010; Wass et al., 2013; Zemblys et al., 2018). For this thesis it is not necessary to dive deeper in the underlying differences between the algorithms, and it will be assumed that the responsibility for taking into account the differences between the algorithm lies with the researcher who will be analysing the eye camera data. What is relevant is that the algorithms will generate a list of gaze fixations enriched by the time they were observed and the coordinates of the gaze direction. This obtained list of gaze fixations needs to be annotated. The process of annotating the gaze fixations can be carried out with the help of proprietary software distributed by eye tracking hardware suppliers or open source solutions.

2.1.2 Proprietary analysing software

Most wearable eye trackers come with proprietary analysing software distributed by their manufacturers. An example of this analysing software is Tobii Pro Lab. The different software tools provide researchers with tools to:

- Replay the recording of the 'scene' camera with the gaze directions from the 'eye' camera overlaid on the recording.
- Export relevant metrics about the amount and characteristics of saccades and fixations.
- Create visualisations that help convey the results of eye tracking experiments. (See figure 6)

Figure 6: Example of visualisation in the form of a heatmap, to show which regions are being looked at the most. *Image obtained from tobiipro.com*

A downside of these proprietary software solutions is that they are expensive and don't allow for much freedom in their functionalities. This limits the accessibility and applicability of the mobile eye trackers. It also provides the research community with a black-box solution, meaning that the exact workings of the software is unknown to its users. Therefore, open source software solutions are desirable to further improve the possibilities of Mobile Eye Trackers.

2.1.3 Overview of open source solutions

In his master's thesis Jiang (2020) provided an overview of available open source tools for automatically analysing mobile eye tracking data. These solutions, as well as other research on this topic (Batliner et al., 2020; De Beugher, 2016; Kurzhals et al., 2016) provided many different methods to decrease analysis time in mobile eye tracking experiments. The different methods can be grouped into three different approaches. In table 2 an overview of the different approaches on the problem of video annotation of mobile eye tracker is presented. Each approach is explained shortly, its advantages and disadvantages are listed and some examples of relevant literature are provided.

Approach	Advantage	Disadvantage	Related paper
Software that decreases the manual labelling time of individual gaze fixations by optimising user interfaces	· Requires no preliminary work to be operational · Is quicker than manual labelling with proprietary software	· A lot of repetitive work	*Benjamins et al. (2018)*
Automatic labelling of gaze fixations by comparing these fixations to known instances of objects of interest with OpenCV functionalities like SIFT and OBS	· Label once manually, and find many similar instances of that object automatically	· Computational load increases with amount of reference labels · OpenCV functionalities fail when object appearance changes because of lighting or orientation	*Brône et al. (2018), De Beugher (2016), MacInnes et al. (2018), and Toyama et al. (2012)*
Automatic labelling of gaze fixations by using neural networks to inference these fixations to trained models of the objects of interest.	· Quick inference, even with lots of reference labels	· Requires a lot of training data to learn the appearance of an object, so available networks are only trained on common objects	*Batliner et al. (2020) and Wolf et al. (2018)*

Table 2: Different methods for improving data analysis

2.1.4 Conclusion

The third approach of using neural networks seems to deliver the most useful and scalable results, but is limited by the fact the models are only trained on a limited set of common objects. If it is possible to create a training set that is representative for the video material that was captured by the 'scene' camera, while keeping the manual labour minimal, both the advantages of the second and the third approach will be utilised. To learn more about the feasibility of creating these training sets the next section will introduce the topic of image classification.

2.2 Image Classification

This section will go more in-depth in the underlying computer vision techniques that are required to semantically label the contents of digital images and videos. First the area of Image Classification will be introduced, which covers the technique of automatically detecting objects in images and videos. After that, the technique of Object Tracking will be explained.

2.2.1 Introduction to Image Classification

Image Classification deals with the problem of *scene interpretation*; what objects are present in an image? Specific difficulties lie in object outlining and object detection. Object outlining tackles the recognition of distinct shapes in images that should represent objects. The goal of Object detection is to match known objects to the contents of an image.

Image Classification is a computer vision technology that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos, and is part of the Machine Learning research area.

Machine Learning can be defined as a process of building computer systems that can infer a pattern or system by creating probabilistic models from example data (Ayodele, 2010). These probabilistic models can be used to make predictions about unknown situations. In the field of Machine Learning, and specifically its subfield Deep Learning, researchers are tackling the problem of Image Classification by training models on large data sets that autonomously infer features from the data. Subsequently they will match these features to new images presented to the models to find occurrences of known patterns (Lu and Weng, 2007).

2.2.2 Subtasks in Image Classification

According to Li et al. (2018), Liu et al. (2020), and Park and Berg (2015), Image Classification can be divided into three different subtasks:

1. Whole Image Classification
2. Instance Segmentation
3. Object Detection

Figure 7: Different types of Image Classification

The first subtask 'Whole Image Classification' uses the entire image as input, and will classify this image into predefined classes. Looking at Figure 7, this task will take the entire image and have one output: CAT. This method does not look at smaller parts of the image, and is therefore not applicable for aiding in the task of data analysis for Mobile Eye Tracking research.

The goal of the second and third subtasks is to match known classes of objects to different parts of the image. The approach between these two is quite different, but both these methods result in a labelled output of different parts of the image. The next section will elaborate further on these two subtasks and their respective approaches.

2.2.3 Instance Segmentation

Instance segmentation is the task of detecting and delineating each distinct object of interest appearing in an image. This results in a pixel map where every pixel belongs to a certain object in the frame. From Figure 7 it can be seen that this approach will highlight the exact shape of the different animals, as well as the name of the animals. Next to the classification of all the objects in the image, this method will also differentiate the different instances of the same objects within the image (Hariharan et al., 2014). The most common framework for this approach is called Mask R-CNN (He et al., 2017), and an approximation of the detection of different regions of interest by this method can be seen in Figure 8. Instance Segmentation models are trained by providing many examples of annotated images with the objects outlined by an accurate mask. This requires complex training data, outlining the masks of objects in the training data is a very time consuming process.

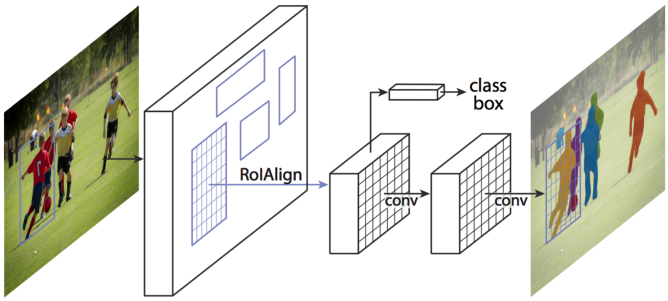

Figure 8: Schematic drawing of the Mask R-CNN framework. *Image obtained from He et al. (2017)*

2.2.4 Object Detection

Object Detection will also detect each distinct object of interest appearing in an image, but does not operate at a pixel level. Instead it repeatedly draws many different rectangles over the image, and tests whether known objects match the content of certain rectangles. Once an object is found within a rectangle, this rectangle is called the bounding box of that object. The most common framework for this approach is called YOLO (Redmon et al., 2016), and a drawing of this process can be seen in Figure 9. Object Detection models are trained by providing many examples of the objects outlined by bounding boxes. Creating training data sets is a lot quicker for this method than it is for Instance Segmentation.

2.2.5 Conclusion

Object Detection is one of the fastest ways to infer the contents of an image. Taking into account the difficulty of annotating training data for Instance Segmentation (it also requires pixel maps as input instead of annotated bounding boxes), Object Detection seems to be the preferred method for annotating the video material from mobile eye trackers.

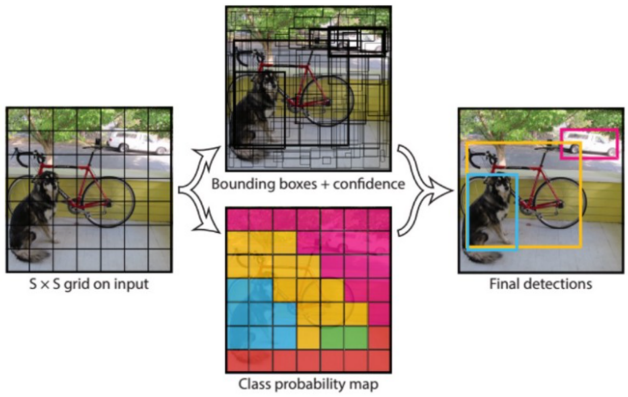

Figure 9: Schematic drawing of the YOLO framework. *Image obtained from Redmon et al. (2016)*

2.3 Using Object Detection to reduce manual annotation time

In earlier research Object Detection techniques were proposed as a solution to the problem with manual video annotation (Brône et al., 2018). In their research, Brône et al. (2018) trained a model that was able to detect persons and faces. They used this technique to automatically label the video material generated by mobile eye tracking experiments. Their results looked quite promising in terms of accuracy and speed.

2.3.1 Problems with Object Detection

Although applying the technique of Object Detection showed some positive results, there is still a problem that needs to be addressed. Training Deep Learning models requires a huge amount of training data, so the technique is very useful when dealing with objects that are very common, like persons or cars, where a lot of annotated data sets are available that can be used to train these models (Marcus, 2018). However, in many cases it is presumed that the areas of interest in mobile eye tracking experiments may not be common objects, but are very specific objects like products in the supermarket, a shopping window or an interface. For these specific objects there are no annotated data sets available to train the object detection models on, so either the technique of object detection is not applicable or an extensive data set of annotated instances of the specific objects should be created. Creating a data set with the techniques currently available still requires a lot of manual labour, which results in this technique not solving the problem introduced in section 1.1.4, but merely shifting the problem at hand to another task.

2.3.2 Object Tracking as a solution

With the current state of technology it seems to be impossible to completely remove the necessity of manual annotation. The goal in improving the process of automatic analysis of video material should therefore be to reduce annotation time for the researcher as much as possible. The question that remains is: Can a new annotation technique create a data set, that can be used to train an object detection model, in less time than it takes to manually annotate the entire video material?

One technique that seems promising to aid in this reduction is object tracking (Sun et al., 2017). With current techniques annotation of objects in videos happens manually frame by frame, which takes a lot of time and requires a lot of repetitive manual labour. With Object tracking, only one initial manual annotation is required. The tracking algorithm will adjust this annotation according to the movements of the object in the screen frame by frame. This alleviates the task of the annotator to manually move the annotation every frame, and it is therefore presumed that the amount of time that is spend annotating the other frames will drastically decrease.

Object Tracking is not a solution to the general problem of extensive manual annotation of video material in mobile eye tracking research, since it would still be necessary to label all the different videos of the individual participants. However, all the annotated frames that are created by the object tracker may be used to overcome the previously mentioned problem of the large amount of data that is required for training object detection models.

2.3.3 Conclusion

Tracking objects in a video generates a lot of data that can be used to train an object detection model, without requiring unreasonable amounts of manual labour. The trained model can then be used to find instances of objects in all the video material produced by the mobile eye tracking experiment.

3 The AnnoTracker - New Proposed Approach

With the knowledge obtained from the previous section a novel approach on the problem of extensive manual annotation will be proposed in this section. First, this novel approach will be introduced conceptually by illustrating the proposed new workflow. After that the actual technical implementation of this approach will be explained.

3.1 Proposed Workflow

The proposed workflow of the new approach consists of three steps. An important assumption in this proposed workflow is that the video material from the 'eye' camera is already processed, resulting in a list of gaze fixations with corresponding time stamps per participant, and that for every participant there is a video of the 'scene' camera that needs to be annotated. This assumption is illustrated in step 0 in Figure 10.

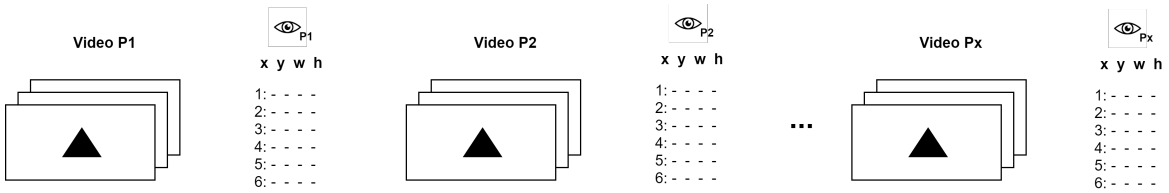

Figure 10: Step 0: After conducting a mobile eye tracking experiment this data is collected. For every participant there is a video of the 'scene' camera, and after processing the video of the 'eye' camera a list of gaze fixations can be obtained.

Step 1 of the workflow (Figure 11) consists of annotating the video material from the first participant or reference video. Using custom software this video shall be opened, where the annotator is able to watch and scroll through the video. At the moment an object of interest is visible, the annotator can draw a rectangle around the object of interest. The video will then continue to play and the tracker will update the location of the rectangle for every frame. The annotator should check that the tracker is working as expected, and adjust where necessary. This should be repeated for every object of interest. The annotated video can then be exported in a format that is usable by the object detection framework.

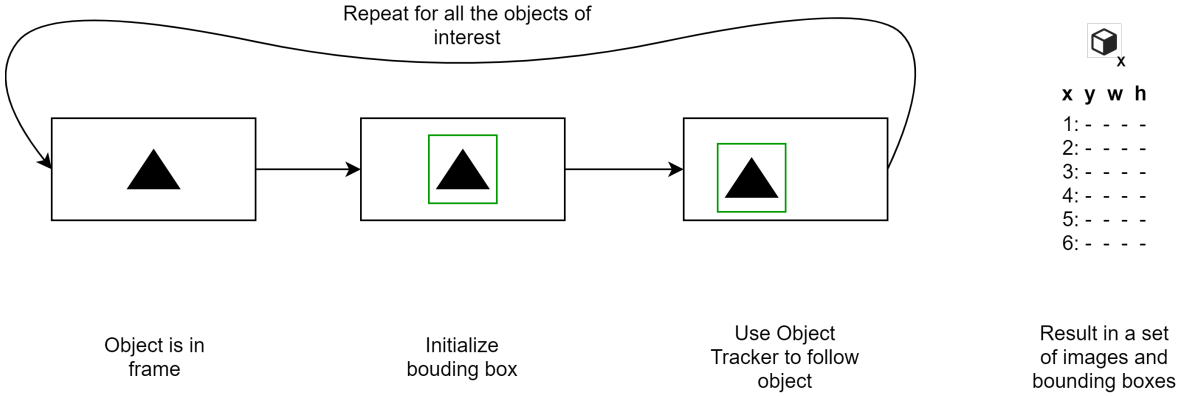

Figure 11: Step 1: Annotating the video of the first participant. Once an object of interest appears in the video, the annotator selects the position in the frame. The object tracker will adjust this annotation in the next frames to the correct position as long as the object is present and the tracker is not lost. This step should be repeated for all object of interest.

Step 2 of the workflow (Figure 12) will take the exported annotations of the first step, and will feed this in a training algorithm of the object detection framework. This algorithm will train a model that is capable of detecting objects that are similar enough to the training data. The training phase will take some computational time, depending on the machine that is used and the amount of annotated data. After the training is finished a model is available for inference on new videos and images.

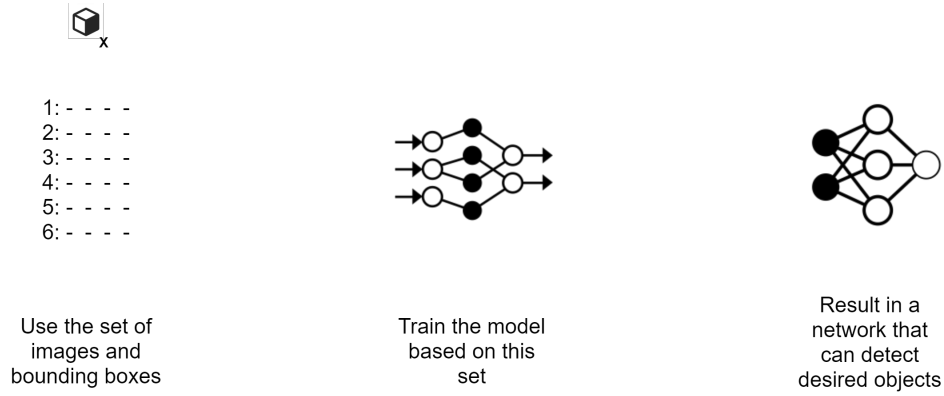

Figure 12: Step 2: Train the neural network. With the annotated data from the step 1 the YOLO object detection model can be trained.

The last step of the workflow (Figure 13) is where the actual annotation of the gaze fixations will happen. For every gaze fixations a timestamp and the coordinates of the fixation are known. The timestamp can be used to find the corresponding frame in the 'scene' camera footage. The trained model from step 2 can take this frame as input and detect any of the objects that it was trained on. If it is successful in finding an object, and the location of the object and the gaze fixation are overlapping to the desired

degree, that fixation can be labelled with the object that was found. Doing this for every fixation will result in a list of annotated fixations.

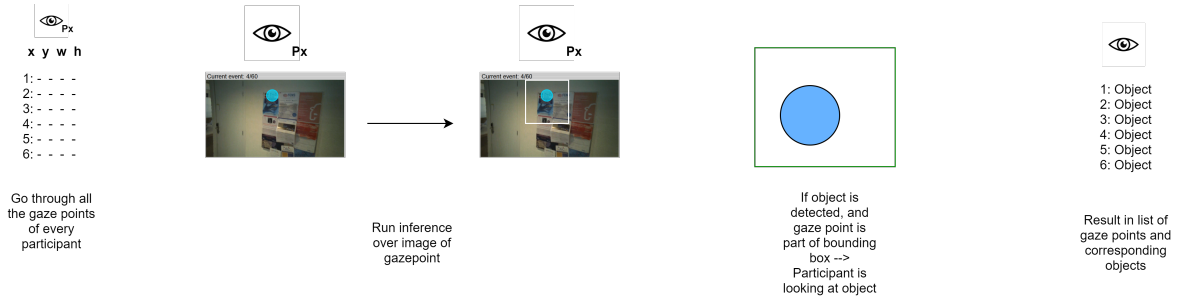

Figure 13: Step 3: Annotate Gaze Points. For every gaze fixation found, the object detection model will label the area that is fixated on. This will result in a annotated list of gaze fixations.

3.2 Technical Implementation

For this new approach custom software was written to allow manual annotation with the object tracker and to export the data to the desired format. The object detection framework that will be used is called YOLOv5 (Jocher et al., 2021). YOLOv5 allows the functionalities of the YOLO framework to be accessed through only a few lines of python code. This usability made the YOLOv5 version attractive for this research. There are more frameworks becoming available rapidly, so choosing which object detection framework is most suited is dependent on what methods define the state of the art at that specific moment. The conceptual workflow should not change depending on which object detection framework is used.

3.2.1 The Tracking Software

The software for annotating the training data with the object tracker was written in Python and the user interface was created with the GUI package PyQt. For the sake of clarity this software will be called the 'AnnoTracker'. The code can be found on Github: https://github.com/Rogions/AnnoTracker. The object tracker that was used for this task is the KCF (Kernelized Correlation Filter) Tracker, implemented in an OpenCV library (Henriques et al., 2014).

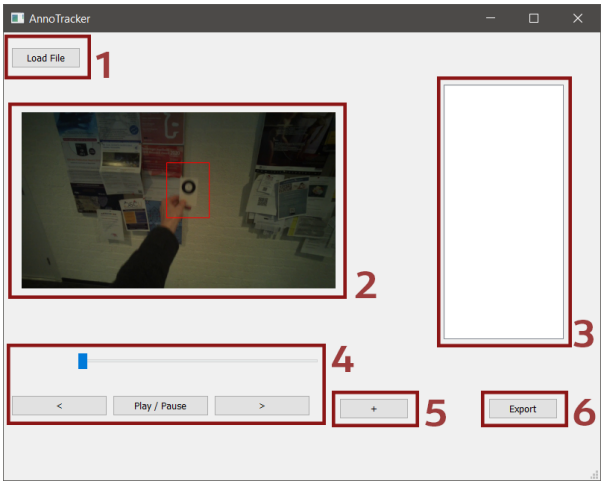

Figure 14: Screenshot of the AnnoTracker software. 1) Load a video of the 'scene' camera. 2) Displays the video and bounding box if available. 3) List of the different objects of interest. 4) Playback controls. 5) Initialise a new rectangle to be tracked. 6) Export the annotations.

Figure 14 shows a screenshot of the 'AnnoTracker' software. Users start by loading a video that needs to be annotated with the 'Load File' button. The video will be displayed, and the playback can be controlled with the buttons and slider beneath the video. When an object appears that needs to be annotated the user will push '+' button and a pop-up screen appears where the user can draw a rectangle around the object with their mouse. After hitting 'Enter', the tracker will be initialised. This tracker will follow the contents of the rectangle until the object disappears from the frame. Users can annotate more than one object by using the list that will appear in the white box. Once a user is satisfied with the annotation, the 'Export' button will save the bounding boxes and images to the correct format. The format that is required for the YOLOv5 object detection model consists of two folders called 'Images' and 'Labels', and a text file naming all the different object classes. In each of these folders two subfolders exist called 'train' (for the training data set) and 'val' (for the validation data set).

For every frame annotated by the 'AnnoTracker' software, that frame is stored as a JPG file in a export folder called 'Images', and the coordinates of the bounding box plus the name of the object class are stored in a text file in an export folder called 'Labels'. The image file and the text file should have the same name so the training algorithm will know which label belongs to which image. The images and labels from the AnnoTracker's export folder should be placed in the correct format required by the object detection framework. In this case this would be the 'train' and 'val' folders. The 'train' and the 'val' folder are used by the training algorithm. The division of annotated frames over the two folders should be somewhere around 70-30, but there is no hard rule on what division delivers the best results.

3.2.2 Training the model

After setting up the required environment for YOLOv5, the annotated files can be used to train the model. The steps of setting up the environment and running the functionalities of YOLOv5 are documented on their repository. After the correct folder and settings for the training model are set up, the

training phase can begin. To understand when training is done, the progress can be visualised and when the functions indicating the process of the training phase flatten out over time generally this means the model isn't improving anymore. After training a '.pt' file is generated, this file contains the weights of the trained model and can be used to detect objects in the next step.

3.2.3 Using the model - Annotating Gaze Fixations

With a trained model the list of gaze fixations can be annotated. The list of gaze fixations should contain timestamps and coordinates of every fixation point. Using the timestamps the corresponding frame of the fixation can be found. This frame can be used as input of the trained model to detect if an object is present in the image according to the model. If an object is deemed present, the model can output the coordinates of the bounding box corresponding to the position of the object. The coordinates of the bounding box can be compared to the coordinates of the gaze fixation, and if they sufficiently overlap the gaze fixation can be labelled with the found object. If no object is detected on the position of the gaze fixation, the unlabelled gaze fixations can be either manually labelled by the researcher, or left empty indicating no object of interest is being looked at.

4 Method

In this section the proposed approach will be compared to manual mapping methods. This comparison should yield some insights in how the proposed approach is functioning in an actual application. These insights can be used to evaluate whether pursuing this approach may be worthwhile.

For the manual mapping method 'GazeCode' (Benjamins et al., 2018) will be used. In their article, Benjamins et al. (2018) compared their software tool to manual annotation with the Tobii Pro Lab software. That comparison will be used as a benchmark for the new proposed approach.

4.1 Setting the benchmark

For the benchmark a video clip of three people playing a card game was used. The video was created with the Tobii Pro Glasses 2, which resulted in a a video clip with the dimensions of 1920 by 1080 pixels with 25 frames per second, for a duration of 330 seconds. The annotators were asked to assign slow phases to areas of interest, with the areas of interest being:

- The player's own hand
- The cards on the table
- The second player on the left
- The third player on the right

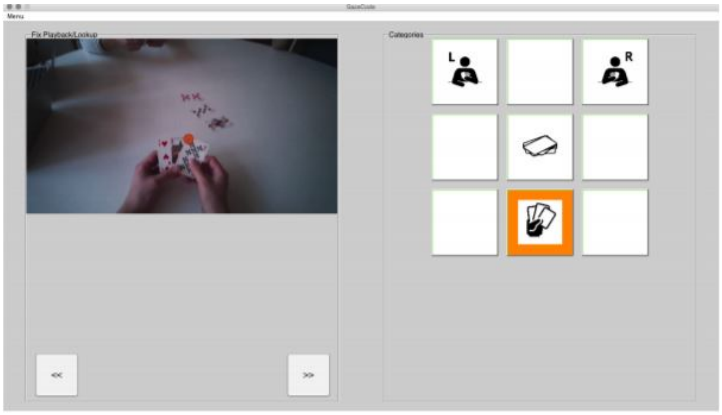

Figure 15: Annotating fixation points with the GazeCode software

The human annotators agreement on the manual mapping was almost perfect with a Cohen's Kappa score of 0.871 and similar across the methods. The annotators that were using the GazeCode software labelled around 0.649 events per second, or for an average of 879 seconds (SD: 124 seconds) for the total duration of the video. The Tobii Lab annotators labelled around 0.292 events per second, for an average of 2880 seconds (SD: 674 seconds).

4.2 Analysing with the new approach

The same clip as the benchmark will be used to train the object detection model. The AnnoTracker software will be used to label the areas of interest in this clip. This labelled data will be used to train an object detection model. For the sake of generalising the results, another clip of the same experiment will be used to run the model on. The results of the model may be skewed if the same video that was used to train the model is also used for the analysis. This is because the model may work very well on the data it was trained on, but perform dramatically if other data is fed into the model. This 'overfitting' can be detected when a second video is used.

4.3 Comparing the methods

The AnnoTracker approach will be compared to the two manual annotations. Important aspects of this comparison are the effectiveness of the object detection model and the efficiency of the whole approach. For this approach to be applicable in other experiments, the recall and precision metrics need to be acceptable. This means that the object detection model should be able to detect a high enough percentage of the cases where it should have detected the object, and that enough of the cases where it did recognise an object are in fact actual cases of the object appearing. The time it takes to annotate an object of interest is also important for the comparison, however the time it takes to annotate videos with this approach scales differently than the manual annotation. Where the manual annotators noted that their efficiency decreased when working for longer periods of time, this approach becomes more efficient the longer the videos take.

5 Results

This section will show the results from the process of annotating video material with the new approach, as well as a comparison to a manual method of annotation.

5.1 Annotating using the AnnoTracker

Annotating the areas of interest took around 2280 seconds (38 minutes). Some AOI's were easier to track than others, due to the amount of times they jumped in and out of the frame. After annotation 6622 frames out of the 8333 frames in the video were labelled. To reduce training time, a lot of the frames were removed from the training set. This is based on the assumption that because of the high frame rate a lot of frames will share very similar content. From the 6622 frames 4635 frames were deemed duplicate, 1325 frames were used for training and 662 were used for validating the training progress. These frames were moved to the correct folder structure, and the training phase was initiated.

5.2 Training the model

Training was done on a Lenovo Ideapad L340-15IRH with a dedicated graphics card (GeForce GTX 1050). Training the 1325 frames for 50 epochs took a little over 5 hours. After these 50 epochs the mean Average Precision (mAP) was 0.98, which is deemed very high. Progress of the training phase can be seen in figure 16 and 17.

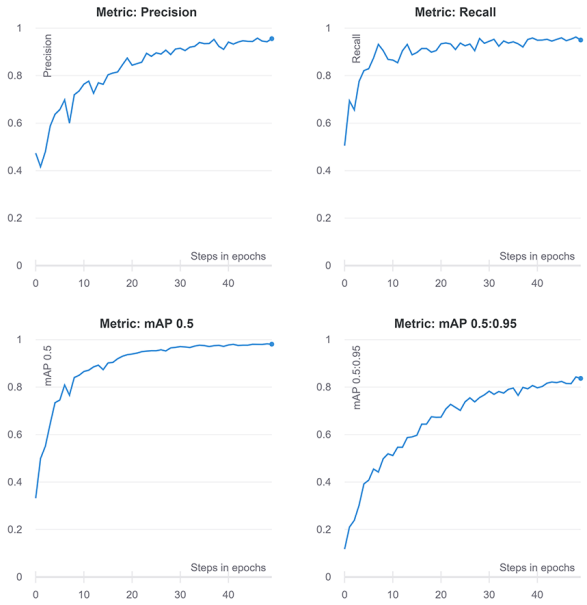

Figure 16: Panels showing the progress of the training phase. The Precision, Recall and mAP metrics should be as high as possible.

Figure 17: Panels showing the progress of the training phase. The Box, Object and Class graphs show the loss functions which should be as low as possbile.

5.3 Testing the model

For the test a second video of the same set up was used. This made sure the model did not only train well on the training data itself, but is also applicable in completely new situations. The test video has a duration of 5 minutes and 32 seconds, or 8305 frames. The areas of interest were also manually labelled in the test video. The observations of the model will be compared to this manual annotation. All the observations can be labelled as either True Positive or a Hit (The AOI was present, and observed), False Positive or False Alarm (The AOI was not present, but was observed), False Negative or Miss (The AOI was present, but not observed) and True Negative or Correct Rejection (The AOI was not present, and not observed). For every AOI the False Alarm Rate *FAR* (False Alarms / (False Alarms + Correct Rejections) and the Miss Rate *MR* (Misses / (Misses + Hits) will be calculated

5.3.1 AOI 1: The player's own hand

	Hand was observed	**Hand was not observed**
Hand was present	4256 *(1.00)*	0 *(0.00)*
Hand was not present	135 *(0.03)*	4049 *(0.97)*

Table 3: The result of the comparison between the model and the manual annotation on the player's own hand, expressed in absolute number of frames. The numbers in brackets show the relative values of correct vs. incorrect observations.

In the video the player's own hand was visible 32 times, for a total duration of almost 3 minutes. The model got all the 32 times correctly, and labelled 4256 frames correctly. The model got 10 False Positives, but these False Positives only occurred for less than half a second, so only 135 frames were False Positives. The model did not miss any instances of the hands being clearly present so no False Negatives were found. The rest of the frames were correctly labelled as True Negatives. The FAR is 0.03 (135 / 4184) and the MR is 0 (0 / 4256). In Figure 18 examples of correct observations can be found.

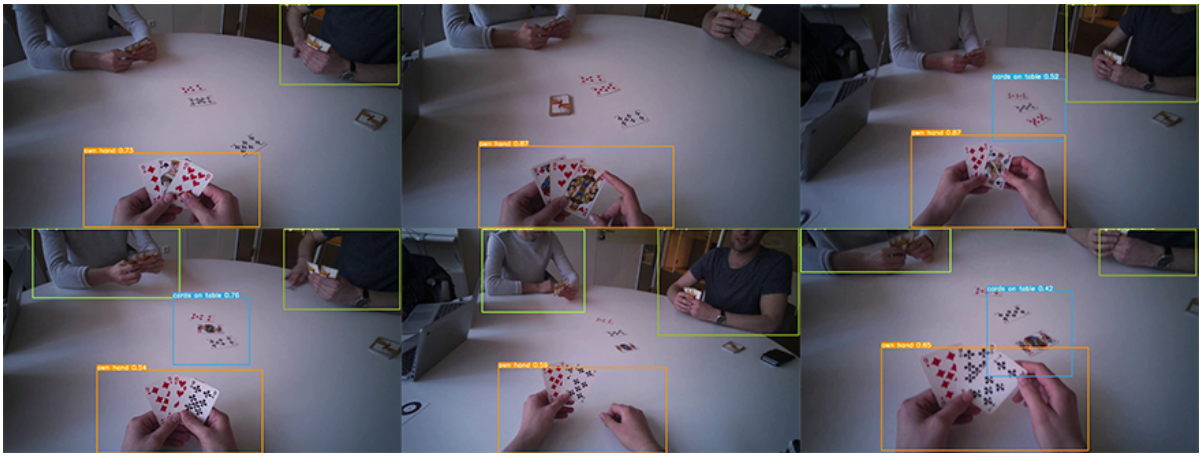

Figure 18: Examples of frames that were correctly identified as player's own hand.

5.3.2 AOI 2: The cards on the table

	CoT were observed	**CoT were not observed**
CoT were present	2260 *(0.49)*	2387 *(0.51)*
CoT were not present	0 *(0.00)*	3658 *(1.00)*

Table 4: The result of the comparison between the model and the manual annotation on the cards on the table (CoT), expressed in absolute number of frames. The numbers in brackets show the relative values of correct vs. incorrect observations.

In the video the cards on the table were visible 29 times, for a total duration of a little over 3 minutes. The model observed 15 of the 29 times correctly, and labelled 2260 frames correctly. The model got no False Positives. The model did miss 14 of the 29 instances of the hands being clearly present which accounted for 2387 False Negatives that were found. The rest of the frames were correctly labelled as True Negatives. The FAR is 0 (0 / 3658) and the MR is 0.51 (2387 / 4647). In Figure 19 examples of correct observations can be found.

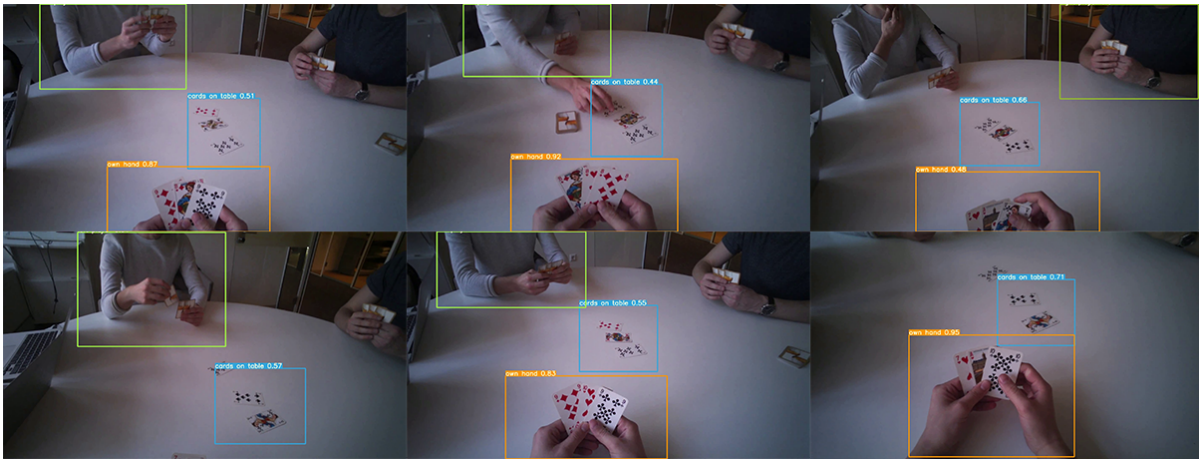

Figure 19: Examples of frames that were correctly identified as cards on the table.

5.3.3 AOI 3: Left player

	LP was observed	**LP was not observed**
LP was present	4208 *(0.67)*	2073 *(0.33)*
LP was not present	50 *(0.02)*	2024 *(0.98)*

Table 5: The result of the comparison between the model and the manual annotation on left player (LP), expressed in absolute number of frames. The numbers in brackets show the relative values of correct vs. incorrect observations.

In the video the left player was visible for a total duration of a little over 4 minutes. The model labelled 4208 frames correctly. The model got 50 frames of False Positives. The model did miss 2073 frames of the left player being clearly present which accounted for 2073 False Negatives that were found. The rest of the frames were correctly labelled as True Negatives. The FAR is 0.02 (50 / 2074) and the MR is 0.33 (2073 / 6281). In Figure 20 examples of correct observations can be found.

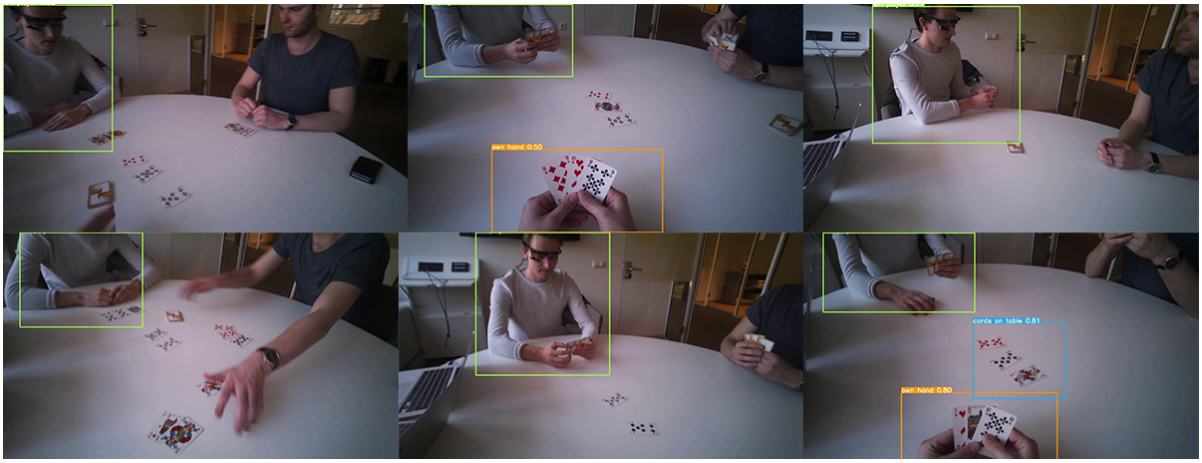

Figure 20: Examples of frames that were correctly identified as the left player.

5.3.4 AOI 4: Right Player

	RP was observed	**RP was not observed**
RP was present	4467 *(0.66)*	2333 *(0.34)*
RP was not present	0 *(0.00)*	1505 *(1.00)*

Table 6: The result of the comparison between the model and the manual annotation on the right player (RP), expressed in absolute number of frames. The numbers in brackets show the relative values of correct vs. incorrect observations.

In the video the right player was visible for a total duration of almost 4 and a half minutes. The model labelled 4467 frames correctly. The model got no False Positives. The model did miss 2333 frames of the left player being clearly present which accounted for 2333 False Negatives that were found. The rest of the frames were correctly labelled as True Negatives. The FAR is 0 (0 / 1505) and the MR is 0.34 (2333 / 6800). In Figure 21 examples of correct observations can be found.

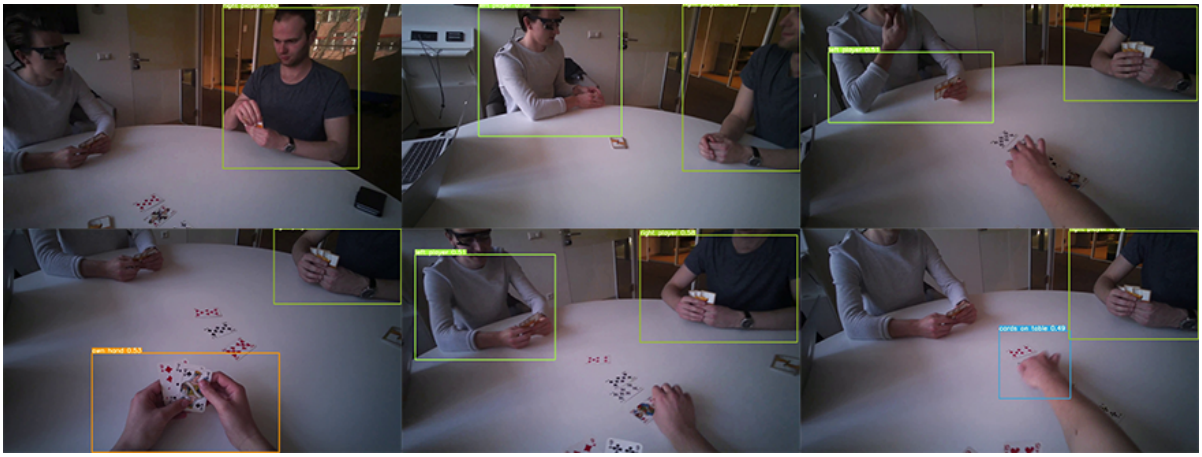

Figure 21: Examples of frames that were correctly identified as the right player.

5.4 Comparison of the methods

The most important factor to remain the integrity of the results is to reduce the amount of false positives. If during the automatic annotation process too many fixations are labelled with the wrong object classes, the results of the experiment can be invalidated. The false positive rate is 185 wrongfully labelled frames divided by the 8305 total frames, resulting in a rate of 0.02 or 2.2%.

The rate of success differs a lot between the AOI's. For the cards on the table the success rate was 0.48 (2260/4647) while for the player's own hand it was 1 (4256 out of the 4256). The left and right players were labelled with a success rate of 0.67 (4208/6281) and 0.66 (4467/6800). If the fixations were to be equally distributed over the four AOI's, the success rate of labelling a fixation would be 70%.

With the information about the initial annotation time and the success rate a formula for the expected time of manual annotation can be derived. The formula consists of two parts, on one hand the initial annotation time to gather training data and on the other hand the amount of time that it takes to manually annotate the remaining unlabelled events. The initial annotation phase in this experiment took 2280 seconds, per AOI this is $2280/4 = 570$ seconds. This seemed to be enough time to get enough training data, so the initial annotation time can be calculated by multiplying the amount of AOI's by 570 seconds. The labelling of the remaining events can be done using the GazeCode software. Using this software, human annotators took around 1.57 seconds per event. The formula to calculate annotation time with the AnnoTracker approach is therefore:

$$AnnotationTime = 570 * nrAOI + 1.57 * 0.3 * x$$

This formula can be compared to the annotation times of the GazeCode method and the Tobii Pro Lab method. This comparison can be seen in figure 22. This comparison does not take into account the mental fatigue of annotating gaze fixations over a longer period of time. Based on this comparison the AnnoTracker approach is faster than the Tobii Pro Lab method after 722 events, and faster than the GazeCode method after 2075 events. If 560 events per participant is a representative number for the rest of the participants, in this experiment the new approach would be quicker than the Tobii Pro Lab

after the second participant, and quicker than the GazeCode method after the fourth participant. It should be noted that this formula is based on the observations from this comparison only, so it does not hold true for any and all cases. It is included in this comparison to visualise the estimated differences between the methods.

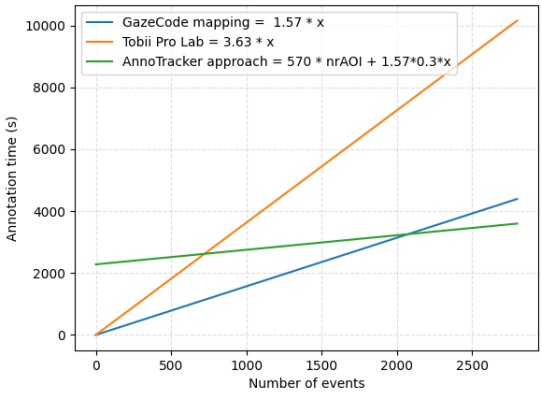

Figure 22: Annotation times for the three different methods. The AnnoTracker approach is quicker than the Tobii Pro Lab approach after 722 events, and is quicker than the GazeCode approach after 2075 events.

6 Discussion

It is one thing to propose a solution for a certain problem, but it is an entire different thing to incorporate this solution in the lives of the people who are facing the problem. As time was a constraint in the process of writing this thesis, not all the questions and hurdles that prevent the AnnoTracker approach from being incorporated in the toolkit of the research community could be addressed. What this thesis did do, is provide the research community with a new route to take in the journey towards a solution for the problem of extensive manual labour in analysing video data from eye tracker experiments. The next sections will cover strengths and weaknesses of the AnnoTracker approach, and will provide ways for further evaluation and improvements.

6.1 Evaluation of the AnnoTracker

Firstly, the possibility to train custom object detection models for the annotation of mobile eye tracking material, using object tracking algorithms to prevent extensive manual annotation, is novel and seems promising. Earlier approaches by other researchers were either able to detect uncommon objects while not allowing a lot of flexibility in the way they appeared in the video material, or only able to detect common objects. This approach combines the benefits of both approaches.

What makes the object detection models more useful than the OpenCV functionalities is their ability to generalise information from the training data. The OpenCV functionalities would not have been able to recognise any of the player's own hands in the comparison to manual annotation. The fact that the object detection model was able to recognise combinations of cards that it had not seen before is truly remarkable, and very promising for the application of object detection models in a large number of situations.

The usage of object tracking for data gathering is interesting because this does, to the best knowledge of the researcher, not seem to happen for generating training data sets at all. This approach may therefore also be useful for other situations where object detection is not yet possible due to the lack of training data sets.

In conclusion, once mobile eye tracking experiments reach a certain scale, and the assumption that the AOI's are trainable is met, this new approach shows a decrease in annotation time compared to complete manual annotation. The field of mobile eye tracking research may benefit greatly from this reduction in annotation time in large scale experiments.

6.1.1 Interpretation of the GazeCode comparison

What should be noted is that the comparison to the GazeCode method was not a conclusive evaluation of the workings of the AnnoTracker approach. It allowed the researcher to observe the AnnoTracker in a 'real life' situation, and did provide the researcher with an estimation whether further improving on this approach is worthwhile. For this comparison an experiment that was already carried out was used, allowing no freedom in the experimental set up; an unfavourable condition for testing this approach. While this may have affected the results negatively, the results may only improve in a condition that is optimised for this approach. With this in mind, the results from the experiment are deemed promising by the researcher. There are multiple ways in which the results could be improved if more control over the experimental set up was possible, which will be discussed in an upcoming section.

6.1.2 Burdening the Computational Effort

It is worth mentioning that the computational time that is required by this approach is still pretty substantial, training machine learning models costs a lot of computational resources. This can be overcome by increasing the computational power available, but this may not always be an option. This limitation is deemed surmountable, because the approach still allows the researcher to spend their time on other tasks. However, it should still be noted that during this computational time the analysis of the video material does not progress.

6.1.3 The Black Box problem

Another limitation is the fact that the results on accuracy of the model can only be determined after the training of the model has completed. A lot of time is still wasted if the researcher finds out that the model does not function as desired, only after completing the preliminary work that is required to get an object detection model to operate. However, after seeing multiple examples of this approach, a researcher does get a better intuition for the probability of success based on the input. This may allow a researcher to determine at an earlier stage that their work is not going to deliver the desired results and opt for another approach.

6.1.4 When is the AnnoTracker approach not applicable?

While testing the AnnoTracker approach on multiple situations, it was found that some situations do not lend themselves as well to the task of object detection. For example, the technique of object detection was also tested on video material that was recorded in a canteen in a university building, and the object detection model failed miserably because items in the canteen were moved, restocked or sold out. Also changes in the lighting condition, may affect performance greatly. It may be for these extreme variable situations that either more annotation in different video material is required, or that using this approach is not beneficial compared to manual annotation. In more 'static' environments, where object appearance does not change much and the amount of possible viewing points of the object is limited, object detection seems to deliver the best result.

6.2 The Future of the AnnoTracker

The AnnoTracker approach of using an object tracker to gather data that can be used to train an object detection model for automatic labelling of AOI's in Mobile Eye Tracking experiments is deemed promising. But before this approach is ready to be incorporated in the toolbox of the research community, first some further research on this approach is required. This section will provide some suggestions for further research to improve the AnnoTracker approach.

6.2.1 Increasing the quality of the automatic annotation

The quality of the automatic annotation is defined by the mistakes, the amount of false alarms and misses, it produces. False alarms are extremely undesirable, because they result in wrongly annotated AOI's and may greatly affect the outcome of the experiments. The AnnoTracker approach did not seem to result in many false alarms (with a FAR of around 0.02), which is a positive indicator of the quality of the automatic annotation.

In the ideal situation, the line of the AnnoTracker approach in figure 22 should be flat, indicating that the approach only consists of an initial set up time and is invariant to the amount of events. This would mean there are no misses; every instance of an object is detected. During the examination of the AnnoTracker approach, it was mentioned by an expert in the field of eye tracking research (I. Hooge, personal communication, 09 July, 2021) that a miss rate of around 5% would be deemed acceptable for the AnnoTracker approach to be feasible in actual research; experimental psychologists rather work long and hard for 95% accuracy than rely on a tool that provides them with 70%. The question remains how the line could be flattened out; or how the amount of misses could be reduced.

6.2.2 Using 'Temporal Smoothing' to decrease the amount of misses

One weakness of the AnnoTracker approach that relates to the amount of misses is that every frame is observed individually while annotating the AOI's. To illustrate this weakness, consider three similar consecutive frames. The AnnoTracker approach may find an occurrence of an object in the first and last frame, but misses the occurrence in the middle frame. If it could use the surrounding frames it may be able to detect its own miss in the middle frame. Using a technique that looks at surrounding frames, often referred to as 'temporal smoothing', a lot of misses may be filled in. Whether this technique would show improvements in annotation quality, and what specific implementation of temporal smoothing yields the best results should be researched. The next two paragraphs will delve into other aspects that may increase annotation quality.

6.2.3 Decreasing the time of initial annotation while improving its quality

In this study the annotation to gather training data was done on the first participant video. What could be done to increase the quality of training data is to create a video in the experimental setting where the objects are filmed from all their possible viewing points, while keeping them in the frame. It is assumed this will decrease the initial annotation time because the object tracker will work better if the object does not jump in and out of the frame, and that the quality of the training data set will improve and cover enough different viewpoints. Further research could prove the assumption that this extra video will indeed improve the initial data gathering time and the effectiveness of the object detection model.

6.2.4 Overcoming the variety of experimental set ups

Another suggestion for future work is that it may be desirable to formalise which situations are most suitable for automatic annotation. It should be researched what characteristics a situation should possess in order to accommodate the AnnoTracker approach. This will allow researchers to make a better estimation of the rate of success of this approach for their specific situation. It would be interesting to see if complete control over the experimental set up, for example by only including high contrast items in the environment or providing perfect lightning conditions, will significantly improve the quality of the automatic annotation.

6.2.5 Overcoming the Black Box Problem

In order to overcome the limitation of wasted time on setting up the object detection model with wrong inputs, the requirements for what constitutes good training data may be formalised. If general rules can be found that can assess the quality of the training data and the variety of data it needs to be able to handle beforehand, the success rate of setting up the object detection model can be found in

advance. This work could also look at a formalisation of when parts of the training data may be deemed duplicate, and what division of the training and validation sets is optimal.

6.2.6 The quality of the Mobile Eye Trackers

A last remark about something that is out of the hands of the researcher is the quality of the video material from the mobile eye tracker. Higher frame rates and higher pixel densities will definitely increase the quality of the object tracker and the object detection models. One thing that was noticed is that the object tracker made the most of the errors when objects were on the outside of the frame, where a dark vignetting can be spotted in the video material. This vignetting may cause problems to the object detection model, but this is not explicitly researched. With the great quality of camera's that are being developed, it is expected that mobile eye trackers that will be released in the future will cause less and less errors because of the data quality. It is assumed this improvement in data quality will increase the effectiveness of the entire AnnoTracker approach.

6.3 The verdict on the AnnoTracker approach

If and how this new approach will be further improved upon will not be decided by the researcher. The researcher may however express its beliefs in the possibilities that were shown in this thesis. The problems that are still being faced are no small hurdles, but the combination of techniques that is proposed seems like a good way forward. The earlier proposed solutions to the problem of extensive manual labour in analysing video data from eye tracker experiments all seem to have hit a glass ceiling, whereas this approach has not reached its full potential yet. It would be interesting to see where this approach could end up once the major issues that are still present would be resolved.

7 Conclusion

This research tried to decrease manual analysis time of mobile eye tracking experiments by developing a technique for semi-automatic annotation of the video material. First, different approaches proposed by other research were analysed to find possible improvements in the current solutions. The technique of object detection models seemed most promising, but a big problem with this technique are the extensive training sets that are required to train these models. This research proposes a new way to annotate a small part of the video material, which can be used to create the required training sets with less manual effort than in traditional annotation tools. After training an object detection model a large set of the fixations points found can be automatically labelled. In the comparison with other approaches the model trained for the comparison was able to label around 70% of the AOI's in the video material. This new approach becomes feasible after a couple of participants, when the initial time it takes to gather the training data weighs up to the manual annotation of the gaze fixations. Complete automatic annotation is still unreachable at this point in time, but this approach does decrease manual annotation time over other approaches with the freedom of training on custom objects.

8 Epilogue

I would like to conclude this thesis by taking a little moment to reflect on the process of writing this thesis and to express an appropriate word of thanks.

I was lucky to be inspired for this topic by a lecture from Ignace Hooge in the opening course of the Master's program Applied Cognitive Psychology. Combining my background in Information Science with the field of Cognitive Psychology was one of my main interests while pursuing this Master's program, and this thesis allowed me to do just that. Developing my skills in programming and performing research in a place outside my comfort zone allowed me to grow tremendously throughout the last couple of months. In a year that was largely dominated by the weird situation the world had found itself in, I think I made good use of the time that was given to me. I'm not sure what the future will hold for this thesis, or for me personally for that matter, but I am nonetheless proud of the work that I've put in and of the thesis it resulted in.

I would like to express my gratitude towards Ignace Hooge for supervising me throughout the last 6 months. Through his remarks and thought-provoking questions I managed to pin the problem down to its essence, and improve the proposed solution beyond what I've could have done myself. It was a shame that we weren't able to physically meet in this process, I'm sure that would have resulted in many more interesting discussions. I'd also like to thank my partner Kristi Bergman for her great support during these months. She was always there to listen to the issues I was facing and helped me greatly in solving them. Because of the work-at-home situation we found ourselves in, she involuntarily became my office buddy, and I couldn't have wished for a better one to spend the countless days at home with. Lastly, I also like to thank the many friends that wanted to listen to me ramble on about my thesis, and for the (sometimes) great feedback they provided.

Rogier Simons
02-07-2021

References

Asman, H., Amin, M., & Wibirama, S. (2019). Exploring the subconscious decision making in neuromarketing research using eye tracking technique. *Journal of Advanced Manufacturing Technology (JAMT)*.

Ayodele, T. O. (2010). Machine learning overview. *New Advances in Machine Learning*, 9–19.

Batliner, M., Hess, S., Ehrlich-Adám, C., Lohmeyer, Q., & Meboldt, M. (2020). Automated areas of interest analysis for usability studies of tangible screen-based user interfaces using mobile eye tracking. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, *34*(4), 505–514. https://doi.org/10.1017/S0890060420000372

Benjamins, J. S., Hessels, R. S., & Hooge, I. T. (2018). Gazecode: Open-source software for manual mapping of mobile eye-tracking data. *Proceedings of the 2018 ACM symposium on eye tracking research & applications*, 1–4.

Bojko, A. (2013). *Eye tracking the user experience: A practical guide to research*. Rosenfeld Media.

Brône, G., De Beugher, S., & Goedemé, T. (2018). Automatic analysis of in-the-wild mobile eye-tracking experiments using object, face and person detection.

Browning, M., Cooper, S., Cant, R., Sparkes, L., Bogossian, F., Williams, B., O'Meara, P., Ross, L., Munro, G., & Black, B. (2016). The use and limits of eye-tracking in high-fidelity clinical scenarios: A pilot study. *International emergency nursing*, *25*, 43–47.

De Beugher, S. (2016). Computer vision techniques for automatic analysis of mobile eye-tracking data.

Eghbal-Azar, K., & Widlok, T. (2013). Potentials and limitations of mobile eye tracking in visitor studies: Evidence from field research at two museum exhibitions in germany. *Social science computer review*, *31*(1), 103–118.

Harezlak, K., & Kasprowski, P. (2018). Application of eye tracking in medicine: A survey, research issues and challenges. *Computerized Medical Imaging and Graphics*, *65*, 176–190.

Hariharan, B., Arbeláez, P., Girshick, R., & Malik, J. (2014). Simultaneous detection and segmentation.

Harwood, T., & Jones, M. (2014). Mobile eye-tracking in retail research. *Current trends in eye tracking research* (pp. 183–199). Springer.

Hasanzadeh, S., Esmaeili, B., & Dodd, M. D. (2016). Measuring construction workers' real-time situation awareness using mobile eye-tracking. *Construction Research Congress 2016*, 2894–2904.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. *Proceedings of the IEEE international conference on computer vision*, 2961–2969.

Henriques, J. F., Caseiro, R., Martins, P., & Batista, J. (2014). High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, *37*(3), 583–596.

Hessels, R. S., Niehorster, D. C., Kemner, C., & Hooge, I. T. (2017). Noise-robust fixation detection in eye movement data: Identification by two-means clustering (i2mc). *Behavior research methods*, *49*(5), 1802–1823.

Holmqvist, K., Nyström, M., Andersson, R., Dewhurst, R., Jarodzka, H., & Van de Weijer, J. (2011). *Eye tracking: A comprehensive guide to methods and measures*. OUP Oxford.

Jacob, R. J., & Karn, K. S. (2003). Eye tracking in human-computer interaction and usability research: Ready to deliver the promises. *The mind's eye* (pp. 573–605). Elsevier.

Jarodzka, H., & Brand-Gruwel, S. (2017). Tracking the reading eye: Towards a model of real-world reading.

Jiang, Y. (2020). *A solution to analyze mobile eye-tracking data for user research in gi science* (Master's thesis). University of Twente.

Jocher, G., Stoken, A., Borovec, J., NanoCode012, Chaurasia, A., TaoXie, Changyu, L., V, A., Laughing, tkianai, yxNONG, Hogan, A., lorenzomammana, AlexWang1900, Hajek, J., Diaconu, L., Marc, Kwon, Y., oleg, ... Ingham, F. (2021). *ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations* (Version v5.0). Zenodo. https://doi.org/10.5281/zenodo.4679653

Karthikeyan, S., Jagadeesh, V., Shenoy, R., Ecksteinz, M., & Manjunath, B. (2013). From where and how to what we see. *Proceedings of the IEEE International Conference on Computer Vision*, 625–632.

Kurzhals, K., Hlawatsch, M., Seeger, C., & Weiskopf, D. (2016). Visual analytics for mobile eye tracking. *IEEE transactions on visualization and computer graphics*, *23*(1), 301–310.

Larrazabal, A. J., Cena, C. G., & Martınez, C. E. (2019). Video-oculography eye tracking towards clinical applications: A review. *Computers in biology and medicine*, *108*, 57–66.

Larsson, L., Nyström, M., Andersson, R., & Stridh, M. (2015). Detection of fixations and smooth pursuit movements in high-speed eye-tracking data. *Biomedical Signal Processing and Control*, *18*, 145–152.

Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y., & Sun, J. (2018). Detnet: Design backbone for object detection. *Proceedings of the European conference on computer vision (ECCV)*, 334–350.

Liu, Y., Song, G., Zang, Y., Gao, Y., Xie, E., Yan, J., Loy, C. C., & Wang, X. (2020). 1st place solutions for openimage2019–object detection and instance segmentation. *arXiv preprint arXiv:2003.07557*.

Lu, D., & Weng, Q. (2007). A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing*, *28*(5), 823–870.

MacInnes, J. J., Iqbal, S., Pearson, J., & Johnson, E. N. (2018). Mobile gaze mapping: A python package for mapping mobile gaze data to a fixed target stimulus. *Journal of Open Source Software*, *3*(31), 984.

Majaranta, P., & Bulling, A. (2014). Eye tracking and eye-based human–computer interaction. *Advances in physiological computing* (pp. 39–65). Springer.

Marcus, G. (2018). Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*.

Mele, M. L., & Federici, S. (2012). Gaze and eye-tracking solutions for psychological research. *Cognitive processing*, *13*(1), 261–265.

Niehorster, D. C., Hessels, R. S., & Benjamins, J. S. (2020). Glassesviewer: Open-source software for viewing and analyzing data from the tobii pro glasses 2 eye tracker. *Behavior research methods*, 1–10.

Nyström, M., & Holmqvist, K. (2010). An adaptive algorithm for fixation, saccade, and glissade detection in eyetracking data. *Behavior research methods*, *42*(1), 188–204.

Park, E., & Berg, A. C. (2015). Learning to decompose for object detection and instance segmentation. *arXiv preprint arXiv:1511.06449*.

Posner, M. I. (1980). Orienting of attention. *Quarterly journal of experimental psychology*, *32*(1), 3–25.

Rayner, K. (1998). Eye movements in reading and information processing: 20 years of research. *Psychological bulletin*, *124*(3), 372.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779–788.

Sharafi, Z., Sharif, B., Guéhéneuc, Y.-G., Begel, A., Bednarik, R., & Crosby, M. (2020). A practical guide on conducting eye tracking studies in software engineering. *Empirical Software Engineering*, *25*(5), 3128–3174.

Sun, X., Cheung, N.-M., Yao, H., & Guo, Y. (2017). Non-rigid object tracking via deformable patches using shape-preserved kcf and level sets. *Proceedings of the IEEE International Conference on Computer Vision*, 5495–5503.

Toyama, T., Kieninger, T., Shafait, F., & Dengel, A. (2012). Gaze guided object recognition using a head-mounted eye tracker. *Proceedings of the Symposium on Eye Tracking Research and Applications*, 91–98.

Wass, S. V., Smith, T. J., & Johnson, M. H. (2013). Parsing eye-tracking data of variable quality to provide accurate fixation duration estimates in infants and adults. *Behavior research methods*, *45*(1), 229–250.

Weibel, N., Fouse, A., Emmenegger, C., Kimmich, S., & Hutchins, E. (2012). Let's look at the cockpit: Exploring mobile eye-tracking for observational research on the flight deck. *Proceedings of the Symposium on Eye Tracking Research and Applications*, 107–114.

Wolf, J., Hess, S., Bachmann, D., Lohmeyer, Q., & Meboldt, M. (2018). Automating areas of interest analysis in mobile eye tracking experiments based on machine learning. *Journal of Eye Movement Research*, *11*(6), 6.

Zemblys, R., Niehorster, D. C., Komogortsev, O., & Holmqvist, K. (2018). Using machine learning to detect events in eye-tracking data. *Behavior research methods*, *50*(1), 160–181.