NTNU
Kunnskap for en bedre verden

Department of Engineering Cybernetics

TTK4550 - Specialization Project

Eye Movement Event Classification Using Low-Cost Commercial-Grade Eye-Tracking Hardware

Author:
Lysberg, Jostein; Hendseth, Sverre

December, 2021

Abstract

This project researches the potential of low-cost commercial-grade eye-tracking hardware using traditional and machine learning methods for automatic eye movement classification. Results are motivated by the possibilities of making informative inferences for performance analytics available for everyone.

Esports is a growing industry, and as kids and adults get progressively into competitive gaming, the need for concrete feedback for performance advancements arises. However, most feedback forms come from direct coaching, which is no reasonable alternative for the casual gamer. May et al. 1990, among others, prove an immense correlation between ocular data and cognitive states. Along with the emergence of low-cost eye-tracking and increasingly advanced machine learning methods, this fact advocates further work towards performance analytics at a low cost and broad availability.

This thesis primarily aims to understand the fundamental properties of a low-cost commercial-grade eye tracker. Such properties were analyzed and tested by applying various methods of eye movement classification. The efficacy of most eye movement classification methods has until now been proved only on data from research-grade eye trackers, which is why their application commercial-grade eye-tracking data is interesting.

Traditionally, methods for eye movement classification have relied on manually coded algorithms. This dependency introduces challenges for the algorithm implementer since every use case requires manually defined parameters, which vary significantly between implementations. Because of this, a machine learning model was also introduced, which could learn such parameters automatically based on a labeled dataset. The need for a dataset was solved by designing a data acquisition pipeline. This pipeline enabled a more efficient process of recording and labeling eye-tracking datasets.

The author found that data from the eye-tracker used, although suffering from some deficiencies, was accurate and consistent enough to be used for most statistical inferences. Additionally, it was found that the data acquisition pipeline could streamline the process of recording and labeling data compared to manual methods.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Eye tracking hardware has been available for a very long time. Studies dating as far back as the 80s show the use of eye trackers in conjunction with computer hardware with accuracies up to about 1/2 degree of visual angle (Ware and Mikaelian 1986). Even before cameras and computers were advanced enough to measure anything accurately, mirrors have been used in conjunction with reading excercises to classify saccades and fixations (Gog and Jarodzka 2013). Current state of the art eye trackers provide massive improvements in accuracy, as well sampling frequency, data quality, data stability and overall ease of use.

Most eye trackers in wide adoption today, are used by researchers for research purposes, with data quality and price tags to match that of large research budgets. As an effect, manufacturers have had limited incentives to promote the commercial availability of the hardware and its accompanying software. Only recently have we seen emerging eye trackers with price tags below thousands of dollars. This enables their use for even the casual gamer, which in turn has remodelled the commercial approach to eye tracking applications.

Traditionally, the leading value proposition for eye tracking hardware has been as an assistive technology for people with disabilities, offering an improvement to the autonomy and quality of life for those in need of alternate input devices (Barry et al. 1994, Corno et al. 2002), but this trend is changing. More recently, the video-game industry has caught wind of the technology, through a series of very promising studies over the past decades (Leyba and Malcolm 2004, Smith and Graham 2006, Tobii 2017). What these studies suggest is that eye tracking hardware need not directly substitute existing control input devices, but could rather serve to complement them. For example, game developers can make game graphics more immersive by letting the user's gaze point determine camera focus, depth of field or light exposure. Game characters may interact differently with the user depending on whether the user is maintaining eye contact and so on. All in all, eye tracking provides a more challenging and immersive experience for the user (Antunes and Santana 2018), and it's adoption is only going to increase as the technology and its applications advances in the near future.

Another exciting use case for eye tracking is driven by the ever expanding competitive gaming environment. As the video game industry is already worth more than the music and movie industries combined (Mangeloja 2019), all estimates show a positive trend in the interest for *Electronic sports* (eSports). In fact, market reports show that the eSports audience reached 474 million people million in 2018, with a year-on-year growth of +8,7% (Newzoo 2021). With this impressive growth comes the ever-increasing demand for competitive performance analytics.

There is always an incentive to be better at whatever game one is playing, especially if the competitive scene is attractive. Naturally the most effective method of improving performance is through direct feedback. Many amateur and pro players tend to subscribe to software services that provide targeted match feedback, as is evident by the success of companies such as Mobalytics and Shadow Esports. At Osiris Performance Analytics AS, we aim to complement existing targeted match feedback with that which can be inferred from eye tracking data analysis.

The analytic potential of eye tracking data is monumental, a potential that exhibits performance enhancing effects on simple learning tasks. The field of psychology amount to years of research on how eye movements and pupil sizes relate to complex cognitive processes (May et al.

1990, Kahneman and Beatty 1966). As Tamara Van Gog, professor of educational sciences at the department of education at Utrecht University states, "Eye tracking is not only a useful tool to study cognitive processes and cognitive load in computer-based learning environments but can also be used indirectly or directly in the design of components of such environments to enhance cognitive processes and foster learning." (Gog and Jarodzka 2013). In a report, she refers to several studies where eye tracking implementations have led to an increase in successful problem solving. Such implementations are outside the scope of this thesis, however they serve as a motivational end to which further research may seek inspiration.

1.2 Research Questions

The choice of eye-tracking hardware depends entirely on the use case, as eye-trackers come with a wide range of properties, advantages, and disadvantages. However, since the author has stressed the aspect of commercial availability, that is naturally going to be the primary deciding factor. The hardware of choice is described in detail in section 4.2. The fact that we sacrifice some data accuracy and quality for availability reveals the first research question which this thesis aims to answer, that is; *"How good is the data quality provided by a commercially available eye tracker, for the purposes of statistical inference by a machine learning model?"*

Since budgets limit research to data gathered from a single eye-tracker, we sadly cannot compare one tracker directly against another. What we can do, however, is to do an in-depth analysis of the data available and discuss possible alternatives where weaknesses are identified. With this in mind, we will answer whether the commercially available eye tracker of choice will be sufficient for further research or if manufacturers still need some years of development before performance analytics by eye-tracking will be open to the general public in the future.

Once hardware limitations have been addressed, we want to measure data quality by a benchmark. As will be covered in section 3.2, researchers have used eye-movement data streams to classify eye movement events for decades through manually coded algorithms and domain experience. The author aims to develop a machine learning model that can contest these established practices by answering; *"Is eye movement data sufficient to train a machine learning model for eye event classification with a level of accuracy surpassing that of manually coded algorithms?"*

Finally, since the classification of pre-defined eye movement events is a problem that inherently requires a machine learning model trained by supervised learning, raw data by itself will likely not be enough to answer the above question adequately. For reasons that will be elaborated in section 2.2, raw data needs to be labeled sample-by-sample with the eye events they represent. This problem brings us to the third and final research question for this thesis; *Is it possible to develop an eye-tracking recording application that can control the environment to such a degree that sample labels can be inferred automatically?*

Chapter 2

Background Theory

2.1 The Oculomotor System

Only second to the brain, the human eye is one of the most complex organs in our bodies. Its anatomy has perplexed scientists for centuries, and the mere fact of its complexity was an essential argument that Charles Darwin had to circumvent during the making of *On The Origin of Species* (Oyster 1999). "If a complex mechanical contrivance like a watch requires a watchmaker, something as wonderful as the eye, which is beyond the wit and ability of a man to create, must have an eye maker."

What Darwin struggled to explain in 1859 is better understood in today's academic community, and it is apparent that vision has proved a winning factor in evolution. Zoologists believe that animals, where eyes have developed beyond a primitive stage, constitute about 96% of known living species. Human eyes are inherited from our primate ancestors, and little has changed in terms of general eye composition since we diverged from our most common ancestors about 30 million years ago. The versatility of the human eye in combination with our brain has been one of the driving forces of our rapid cultural evolution. This thesis is chiefly concerned with the oculomotor system, an intricate network of muscles and neural circuits that control eye position and, in turn, determine how humans perceive their environment.

2.1.1 Brief Anatomy

The oculomotor system can not be explained in detail by its components alone. The human eye incorporates several distinct major and minor structures and regions that transform light into perception. However, its macroscopic anatomy is quite simple and readily explainable. As Gordon Walls, a renowned professor of physiological optics and optometry, once put it: "The eye is (simply) a fluid-filled chamber enclosed by three coats or layers, of tissue. It is an optical system made of leather, water and jelly" (Oyster 1999), as can be observed by figure 2.1.

Leather would refer to the white *sclera*, a rigid tissue making up most of the outer coat of the eye. The gel-like *vitreous* is what makes up most of the interior volume. Light is admitted through and partially refracted by an area at the front of the eye, where the sclera yields to a transparent coat termed the *cornea*. After the cornea, light passes through an aperture stop known as the *pupil*. The pupil is essentially just an opening within the *iris*, a sphincter known for its abundance of melatonin pigments and rich colors. Its purpose is to regulate the pupil opening. Light is further refracted in the *lens* before reaching the *retina* at the very back of the eye. Here, the light is processed by millions of hypersensitive photoreceptor cells and transferred to the brain.

The Extraocular Muscles

Central to the oculomotor system are the muscles that allow for physical translation and rotation of the eye in space, enabling the brain to control which parts of the environment it wants to perceive. Movement is made possible by *extraocular* muscles arranged in three agonist-antagonist pairs, meaning there are six in total, each pulling in the opposite direction of another. Together, they allow for accurate vertical and lateral positioning and -90˚to +90˚torsions. Muscle fibers are inherently elastic, and each extraocular muscle exerts passive force even when not actively

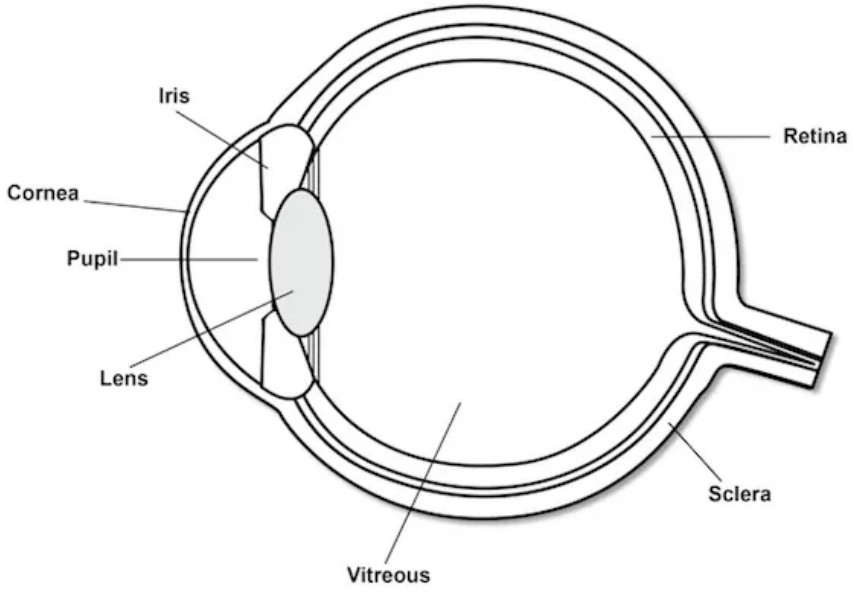

Figure 2.1: Anatomy of the human eye. The model is greatly simplified to include only major structures.

engaged. This elasticity means that for the eye to hold any stationary position, all six muscles are counterbalanced by an equal and opposite force, producing an equilibrium.

2.1.2 Eye Movement

Since most retina photoreceptors are concentrated at the fovea, a minuscule 1.5mm wide pit right behind the lens, only a tiny area in the field of view provides clear vision. Yet, the brain dedicates as much as 25% of the visual cortex to processing information from this little spot (Hubel and Wiesel 1974). Therefore, as we will see, eye movements are crucial to moving the field of view between areas of interest. Other reasons are image stabilization, focus, and more. This importance gives rise to the need for complex eye movement patterns, some of which will be mentioned here.

Whenever attention is shifted from one point to the next, a *saccade* event occurs. These are typically between 30-80ms in duration and can reach velocities of up to 500°/s (Holmqvist et al. 2011). Although the eye is not particularly heavy (7.5g in adults), its mass does pertain to a tiny bit of inertia which resists movement. Additionally, since its primary interior and cornea are made of a gel-like substance, most saccadic events are followed by a "wobbling." These *post-saccadic oscillations* (PSOs) have durations of 10-30ms.

Perhaps one of the most discussed events in eye movement research, along with saccades, are *fixations*. Only broken by other movement events, they occur whenever the user is focusing attention on one particular point. Fixations are usually tens or a few hundred milliseconds in duration but can last for several seconds at most. During this time, the extraocular muscles will move the eye in very small *microsaccades* and *drifts* around the point of fixation. These micro-movements require sensitive equipment to detect and are no more than rapid, high-frequency flicks of about 0.2°of arc. Although seemingly erratic, these movements are critical for vision, as a fully stabilized image on the retina will eventually fade and disappear (Oyster 1999). What we perceive as vision is essentially a series of very rapid after-images, similar to that produced by a camera flash.

One final eye movement type is called the *smooth pursuit* and is in some ways similar to the saccade in how it moves the eye in continuous motion. Functionally, however, the movement is driven by a completely different part of the brain. As its name suggests, it can be classified by much smoother motions than the saccade, with velocities between 10-30°/s and arbitrary durations. They occur only when the eyes actively follow a moving target or conversely when fixating on a stationary object while the head or body is moving. The reader is encouraged to attempt such a

movement themselves, perhaps when looking at a plain, white wall. They are likely to realize that without such "triggers," it is impossible to produce a smooth pursuit forcibly.

2.2 Machine Learning

Artificial intelligence is a field of research that has been practiced for a very long time, contrary to popular belief. The first work that is now recognized as AI was written by McCulloch and Pitts 1943. Even still, the concept of artifacts operating under their own control can be traced back to Alexandria ca. 250 BC, where a water regulator was built that could maintain a constant flow rate. Other examples of self-regulating feedback control systems include the steam engine governor and the thermostat, all invented before the 19th century (Russell and Norvig 2009). This thesis will concentrate on machine learning, a particular form of artificial intelligence that employs prior knowledge of historical data to tackle tasks that are too difficult to solve with fixed programs written and designed by human beings (Goodfellow et al. 2016).

2.2.1 Learning Algorithms

A machine learning algorithm is characterized by the fact that it can learn properties from data. Mitchell 1997 famously defines this aspect of learning as "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." Since machine learning algorithms can be employed in a wide range of problems and trained by an equally wide range of methods, T, P, and E in this definition can be constructed as just about anything. However, a very common class of tasks T central to this thesis is that of *classification*. The experience E required to train towards this task is usually provided through either *supervised-* or *unsupervised learning*.

Classification

In the classification task, the computer program is asked to determine which of k discrete categories some input belongs. Given data (experience E) produced by a function $f : \mathbb{R}^2 \to \{1, \ldots, k\}$, a learning algorithm tasked with classification will generate a hypothesis $h : \mathbb{R}^2 \to \{1, \ldots, k\}$ that approximates f. Given an input vector $\mathbf{x}$, $h(\mathbf{x})$ outputs a probability distribution over possible categories. One popular application is that of object detection, where an image is given as the input $\mathbf{x}$, and the output is the category to which an object in the image belongs. If $k = 2$, the learning problem is called binary classification, and the hypothesis h merely outputs the probability of whether an input represents a single target category or not. Multi-class classification is more common in the general case, however.

Since we often cannot assume any prior knowledge of the inherent properties of f, choosing a hypothesis h with which to approximate it is no trivial task. We say that h is selected from a *hypothesis space* $\mathcal{H}$, which needs to be defined by the learning algorithm designer. Looking at figure 2.2, one can see the importance of choosing a hypothesis space that is complex enough to approximate f accurately, yet simple enough that the function does not overfit. For this example, a 12-degree polynomial is required to produce an approximation that agrees with all the data. However, since we cannot assume that f is not stochastic, this polynomial might generalize very poorly to unseen data. In such a case, a simpler hypothesis in a linear or a sinusoidal function might be the optimal choice.

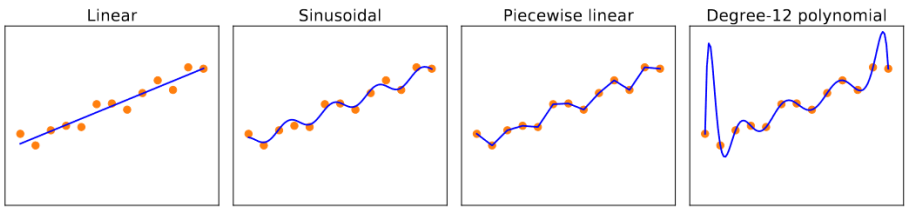

Figure 2.2: Finding hypotheses to fit data. The four plots each show best-fit functions from four different hypothesis spaces trained on the same dataset.

Source: Russell and Norvig 2009

Supervised Learning

A machine learning algorithm can be called a supervised learning algorithm if it is trained by experiencing a dataset of examples $\mathbf{x}$, where each example is associated with a target y. The training process thus becomes to approximate a function that can reproduce y given $\mathbf{x}$.

One classic example is that of linear regression, which is supervised learning in its simplest form (Goodfellow et al. 2016). The goal of linear regression is to predict a target value $\hat{y} \in \mathbb{R}$ from an n-dimensional input vector $\mathbf{x} \in \mathbb{R}^n$. Since we know then that the output should be a linear function of the input, the problem becomes approximating the hypothesis given by equation 2.1. Here $\mathbf{w} \in \mathbb{R}^n$ is a vector of model parameters, which control the behavior of the system.

$$h(\mathbf{x}) = \hat{y} = \mathbf{w}^T\mathbf{x} \tag{2.1}$$

Now, to determine the optimal value of $\mathbf{w}$, we need a performance measure P. For this particular problem, a typical choice is the *mean squared error* (MSE), given in equation 2.2. MSE is calculated from a matrix of m input vectors $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, their corresponding m target values $\mathbf{y} = \{y_1, \ldots, y_m\}$, and model parameters $\mathbf{w}$.

$$MSE = \frac{1}{m}\sum_i(\hat{\mathbf{y}} - \mathbf{y})^2 = \frac{1}{m}||\hat{\mathbf{y}} - \mathbf{y}||_2^2 \tag{2.2}$$

By setting $\hat{\mathbf{y}} = \mathbf{y}$, one can see that $MSE = 0$, and furthermore that it increases linearly with the euclidean distance between $\hat{\mathbf{y}}$ and $\mathbf{y}$. As such, the optimal model parameters can be obtained by minimizing MSE with respect to $\mathbf{w}$. This can be done by solving for where its gradient is $\mathbf{0}$, as is in equations 2.3 to 2.6.

$$\nabla_{\mathbf{w}} MSE = \mathbf{0} \tag{2.3}$$

$$\Rightarrow \nabla_{\mathbf{w}} \frac{1}{m}||\hat{\mathbf{y}} - \mathbf{y}||_2^2 = \mathbf{0} \tag{2.4}$$

$$\Rightarrow \nabla_{\mathbf{w}} \frac{1}{m}||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 = \mathbf{0} \tag{2.5}$$

$$\vdots$$

$$\Rightarrow \mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{2.6}$$

Evaluation of equation 2.6 results in a value of $\mathbf{w}$ which optimally fits the training dataset. As such, it constitutes a simple learning algorithm (Goodfellow et al. 2016).

2.2.2 Decision Tree Classifier

Decision tree induction is one of the simplest and yet most successful forms of machine learning (Russell and Norvig 2009). It is also a supervised learning algorithm, as it requires a dataset of labeled examples to be trained. The Decision Tree Classifier takes a vector of attribute values as input and returns a "decision." It reaches this decision by performing a sequence of tests on one or more attributes. An example is presented in figure 2.3. Here, one can imagine that the species of an animal is to be determined based upon a set of attributes {"Has feathers?", "Can fly?", "Has finns?"}. The likely animal species can be determined by following a path from the root node to a leaf node. Although this example is simple, the general principle holds for more complex classification problems as well.

The training process of decision trees is done with specialized algorithms, the specific details of which are outside the scope of this thesis. However, they are usually based on a greedy divide-and-conquer strategy, where the training dataset is split on whatever attribute is most important. Which attribute this is, and at what attribute value the split is done, is determined by a notion of *information gain*.

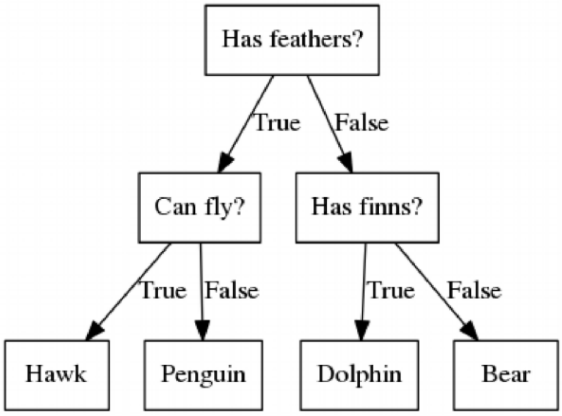

Figure 2.3: Decision Tree Example. Each internal node represents a test on one attribute, and its branches are labeled with possible attribute values. Leaf nodes represent a decision.

2.2.3 Ensemble Learning

Ensemble learning is a relatively straightforward method that has a remarkable effect of reducing generalization error (Goodfellow et al. 2016). When a machine learning model has been trained, one is left with a single hypothesis h that, to a certain degree, is able to approximate some function f. However, if the model is trained once more, the resulting approximation would likely produce slightly differing predictions. The notion of ensemble learning is to train multiple independent models and combine their predictions. Specifically, by determining the output by majority voting or an average, the method is termed *model averaging*. Some ensemble methods construct their ensemble of models by random, others by some rationale. *Bootstrap aggregating* (bagging), for example, trains each model on a subset of the original training dataset by sampling at random with replacement. This strategy ensures an inherent distinction between models, which offers a broader hypothesis space and reduced generalization error.

Chapter 3

Previous Research

3.1 Eye-Tracking Research

As the reader might agree, vision is perhaps the most predominant sense of perception that humans enjoy. We can interpret vast amounts of information every second from the raw signals of millions of optic nerve fibers. This process is so complex that a significant fraction of the cortex is involved (Klatzky and Giudice 2012). As such, it is natural to believe that our eyes are a good metric when studying the internal cognitive functions of our brain. Cognitive psychologists have exploited this fact for over two centuries (Eckstein et al. 2017), with studies in fields ranging from education and marketing to neuroscience and artificial intelligence.

Some studies have even studied the possibility of inferring subject intention from their eye movements. Ballard et al. 1992, for instance, wanted to investigate whether gaze patterns were an indicator that preceded actions. To do this, he conducted an experiment where subjects were to move a set of colored blocks to match the pattern of a given model, while a mobile eye tracker was set up to record their eye movements. He found that there was indeed a strong link between a subject's eye movements and their actions. Gaze followed a clear pattern of checking out a block before picking it up and the model before placing it down. Similarly, Land et al. 1999 measured a subject's eye movements as they performed daily tasks, such as brewing tea or making a sandwich, again finding that the subject's eyes always revealed their intentions before acting.

Eye-tracking research in the field of education has been dominated by its use in investigative reading (Knight et al. 2014). Tracking the reader's saccadic movements between sentences and fixation times has provided profound insights into comprehension activity. Gog, Jarodzka et al. 2009 applied this fact in a study into the attentional processes of expert versus novice readers and found clear contrasts in strategies employed for comprehension. Other studies (Rayner 1998, Castelhano and Rayner 2008) report that expert readers scan pages more quickly, continuously, and consistently, with an ability to concentrate attention on parts of the text that they believe to be critical. On the other hand, novice readers read linearly, displaying frustration and often giving up on the task. Studies such as these prove that eye-tracking is invaluable in assessing comprehension and other cognitive states.

3.2 Traditonal Classification Methods

Eye-tracking research is rarely based on raw data alone. Almost all eye-movement researchers require methods to parse eye movement data and detect different eye movement events, such as fixations, saccades, and smooth pursuits. During the conception of eye-tracking in research, such methods were based on manual labor. Hartridge and Thomson 1948 recounts techniques requiring detailed examination of single frames of film using a low-power microscope, as well as other necessarily complex approaches to be regarded as accurate, which at most could process one frame every two or three minutes. However, with the emergence of computers, eye movement classification has primarily been done algorithmically.

Perhaps the most commonly used algorithms for eye event classification are based on the assumption that fixations points tend to be clustered closely together because of their low velocity. These *Dispersion-Threshold Identification* (IDT) algorithms classify points based on a

point-to-point dispersion metric. One implementation is described by Salvucci and Goldberg 2000 and recited in algorithm 1. Additionally, some IDT algorithms support classification accuracy by the added assumption that fixations typically have a duration of at least 100 ms. IDT algorithms, therefore, require the manual definition of both a dispersion- and a duration threshold, which calls for some domain experience by the designer. Although guidelines for defining such thresholds exist, some form of exploratory data analysis and tuning is required to obtain reliable classifications.

Algorithm 1 Dispersion-Threshold Identification

```
1:  procedure IDT(vec, dispThresh, durThresh)
2:      while vec not empty do
3:          window ← first subset of vec to cover durThresh
4:          if DISPERSION(window) < dispThresh then
5:              while DISPERSION(window) < dispThresh do
6:                  window ← window + next point in vec
7:              end while
8:              for point in window do
9:                  point ← FIXATION
10:             end for
11:             vec ← vec − window
12:         else
13:             vec ← vec - first point in vec
14:         end if
15:     end while
16:     return vec
17: end procedure
18:
19: procedure DISPERSION(window)
20:     return (MAX(window.x) − MIN(window.x)) + (MAX(window.y) − MIN(window.y))
21: end procedure
```

Another commonly used set of algorithms classifies points as saccades by a single point-to-point velocity threshold. As mentioned in section 2.1, saccade movements can gain speeds of up to 500°/s, which is certainly distinguishable from the ideally static fixations. With a single threshold, these algorithms are relatively straightforward to design. One implementation of such a *Velocity-Threshold Identification* (IVT) algorithm is presented in Salvucci and Goldberg 2000 and recited in algorithm 2. As with the IDT algorithms, its parameters require exploratory data analysis and domain experience to determine.

Algorithm 2 Velocity-Threshold Identification

```
1:  procedure IVT(vec, velThresh)
2:      for all points in vec do
3:          Calculate point-to-point velocity
4:      end for
5:      for point in vec do
6:          if velocity > velThresh then
7:              point ← SACCADE
8:          end if
9:      end for
10:     return vec
11: end procedure
```

3.3 Towards Machine Learning

Although this thesis aims to implement a machine learning model that can compete with the traditional algorithms mentioned above, it is not the first to attempt so. Both simple and very advanced machine learning methods have been implemented, usually with significant improvements in both autonomy and accuracy compared to traditional methods. Such algorithms are considered by many the new paradigm in eye movement event classification.

One breakthrough in this regard was made by Zemblys, Niehorster et al. 2017, where a Random Forest Classifier was utilized on a dataset recorded by a 1000Hz EyeLink 1000 eye tracker. Individual samples were labeled manually by an expert with nine years of eye-tracking experience into fixations, saccades, and *post-saccading oscillations* (PSO). To accommodate different sampling frequencies, they augmented the dataset by resampling to 60, 120, 200, 250, 300, 500, and 1250Hz and adding multiple noise levels to each resample. The result was a model that beat traditional classification methods by large margins on the high-frequency dataset clean of noise. On the lowest-frequency dataset, the results were much worse. However, because of the manual features chosen for classification, many of which relied on sample-to-sample resolutions of less than 10ms, downsampled datasets had a significant disadvantage. The 50Hz dataset had sample-to-sample intervals of 20ms, for instance. Because of this, low-cost commercial eye trackers were not adequately represented.

A year later, many of the same authors produced a new model, presented in Zemblys, Niehorster et al. 2018. This model was fully end-to-end and capable of classifying fixations, saccades, and PSOs without the need for manually generated features. That was made possible by a deep neural network consisting of convolutional and recurrent layers. The model was trained on data recorded on a high-end tracker at 500Hz and manually labeled by experts. Results were in near-perfect agreement with manual coders. Recurrent neural networks also made it possible to train a generative network. This network was used to augment the training dataset by generating more of the less occurring labels, such as saccades. Doing so helped alleviate the problem of imbalanced frequency of occurrence of classes, which needs to be accounted for in this thesis.

Chapter 4

Hardware and Data

4.1 Eye Tracking Technology

Eye-tracking systems have existed in some way, shape, or form since the late 1800s (Holmqvist et al. 2011). The first occurrences included bite-bars to ensure still head positions and mechanical rings attached to the eyeball. More modern techniques are based on electromagnetic inductance in a specially constructed lens, others directly on the electromagnetic activation of the extraocular muscles. The latter is more commonly known as electrooculography and is still present today in some systems. Ever since the appearance of the pupil- and corneal reflection method, the nature of which will be explained shortly, that has been the primary approach in all modern eye-tracking systems.

The dominance of the pupil- and corneal reflection method comes from its minimally intrusive manner, as it allows for precise and accurate gaze estimation from a video recording of the user's eye movements. The hardware of such eye-trackers consists of at least one high-resolution and high-frequency camera, accompanied by one or more infrared light sources directed at the user. These light sources will produce a reflection on the cornea, which, together with the pupil, serves as reference points to gaze direction and head position. An example is illustrated in figure 4.1. Positions of reference points are subsequently determined using computer vision methods on images produced by the camera. A calibration process is initially required to provide the system with known reference point relationships corresponding to known gaze points in the tracking area. The rest of the tracking area is then interpolated between calibration points. While even one camera and one light source provide a reasonably accurate gaze position as long as the head is fairly still, more cameras and infrared sources can be used to relax the constraints on head movement and calibration (Holmqvist et al. 2011).

All eye-tracking methods suffer from a deficiency in degrading estimation with large gaze angles. In the extremes, the corneal reflection is often lost. As such, eye-trackers utilizing the pupil- and corneal reflection method rarely support gaze angles beyond 40°in the horizontal direction and 25°in the vertical direction. Given the viewing distance d and the gaze angle θ, the corresponding unit x in the stimulus space is given by equation 4.1. Note, however, that this relationship only holds when θ is small, i.e., when the user looks at points close to the tracker camera. As such, the same visual angle θ_1 may result in different displacements (x_1 and x_2) on the stimulus for progressively larger gaze angles, as illustrated in figure 4.2.

$$\tan \frac{\theta}{2} = \frac{x}{d} \tag{4.1}$$

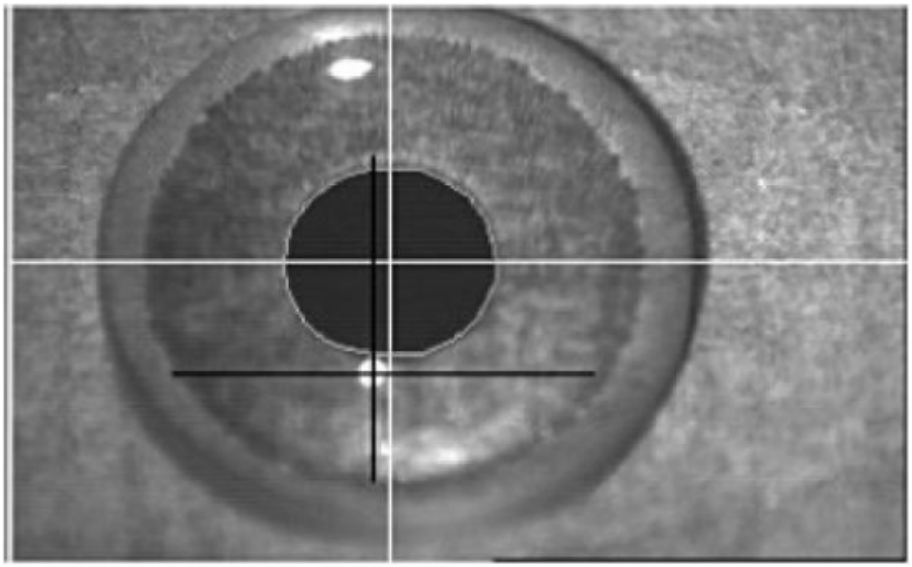

Figure 4.1: Demonstration of pupil- and corneal reflection. The white and black crosses correspond to the point indentified as the pupil and corneal reflection centers, respectively.

Source: Holmqvist et al. 2011

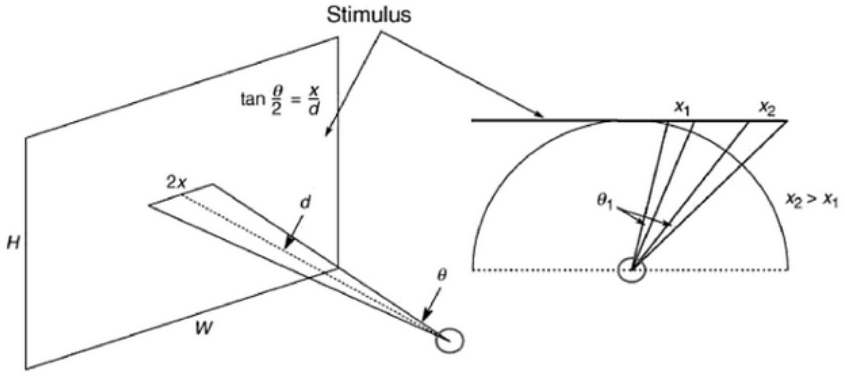

Figure 4.2: Geometric relationship between stimulus and degrees of visual angle.

Source: Holmqvist et al. 2011

All eye trackers come with a given sampling rate, and the speed at which data is recorded is typically the primary contributor to its price. Considering eye movements as an oscillating behavior, one can use the Nyquist-Shannon sampling theorem to argue that a sampling frequency at least twice as large as the largest frequency component of the signal is required. This means a sampling rate of 300Hz is necessary to estimate the movements of micro-saccades, which can oscillate at 150Hz. Still, for most use cases, it is often sufficient only to evaluate the more significant eye movements such as saccades, fixations, and smooth pursuits, which are not oscillating and subsequently require less strict sampling rates.

4.2 Tobii Eye Tracker 5

The eye-tracking hardware used for the entirety of this thesis is the *Tobii Eye Tracker 5* (Tobii ET5), shown in figure 4.3. It is produced by a Swedish company with 20 years of experience, which is currently the only actor in the market to make low-cost eye-trackers. They have had this position since mid-2014. Released in July 2020, Tobii ET5 is their third iteration in low-cost commercial-grade eye-tracking hardware.

Tobii ET5 features two 850nm near-infrared light illuminators in a stereo configuration, roughly 25cm apart. Between these illuminators is an optical biometric face sensor. Together they allow for remote eye tracking without any intrusions on the subject, allowing for free movement within a given head-box. Remote eye-tracking is possible due to the pupil- and corneal reflection method of gaze estimation, supported by head pose estimation by a convolutional neural network. The head-box is defined as roughly 60-80cm from the monitor and about 30cm vertically and laterally. From there, the tracker can estimate gaze in a 35x35cm on-screen track-box. That is enough to cover monitor sizes up to a maximum of 27", with an aspect ratio of 16:9. It records head position at 133Hz. However, gaze estimation is still limited to 33Hz due to a limitation in the illuminator emission frequencies. It is mounted directly below the monitor, as shown in figure 4.3.

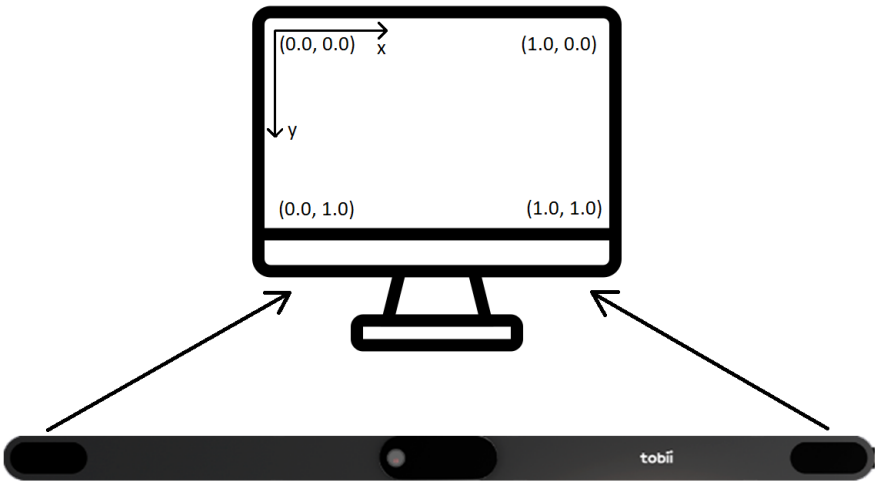

Figure 4.3: Hardware of Tobii Eye Tracker 5, as well as a depiction of its setup.

4.2.1 Data Output

Data from Tobii ET5 is output at 33Hz, in on-screen coordinates by the coordinate system shown in figure 4.3. Coordinates are given in floating-point values from (0.0, 0.0) in the top-left corner to (1.0, 1.0) in the bottom-right corner. Upon installation, between subjects and otherwise as often as possible, a calibration is required to get accurate gaze estimations. There is an average interval of about 30ms between samples in regular operation. The high-frequency camera can detect any blinks by occlusion of the pupil. On such an event, or when gaze leaves the track-box, samples are excluded from the data stream. Once the pupil is again visible, or the subject returns their gaze within the track-box, the tracker requires some time to recover a gaze estimation. As such, intervals of missing data samples are not necessarily suitable measures of actual blink duration.

Additionally, data is lightly filtered to remove noise. Sadly, the details of this processing are not publicly available to the customer for Tobii's low-cost eye trackers.

Chapter 5

Methods

5.1 Data Acquisition Pipeline

Data gathered from Tobii ET5 is processed through a 3-step pipeline, as illustrated in figure 5.1. This *Data Acquisition Pipeline* is required to go from a raw data stream of eye-tracking data to a dataset ordered by timestamps, labeled, and ready for classification. Each step will be detailed further in the following sections.

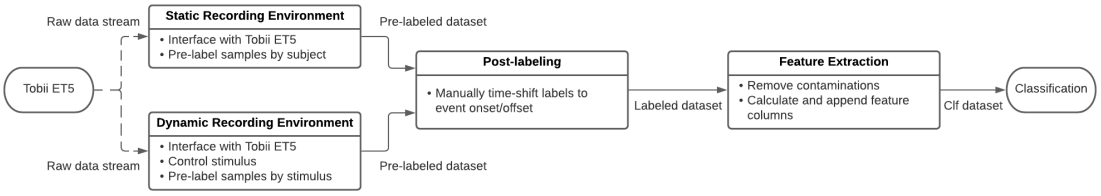

Figure 5.1: Data Acquisition Pipeline. Each box is a step in the pipeline. Dotted lines are data streams, full lines are datasets.

5.2 Recording Environments

This section will explain the first step in the data acquisition pipeline detailed above 5.1, namely the two recording environments. Their function is to interface with Tobii ET5 to extract raw eye movement data and organize it in a structured dataset. Additionally, the dataset is pre-labeled with approximate eye movement event labels to simplify the manual post-labeling process.

Both recording environments are implemented in `.NET Core 3.1` using `C# 8.0`. Interfacing with Tobii ET5 is done with an API-wrapper implemented in `.NET Framework 4.8` using `C# 7.3`.

5.2.1 Static

A scaled-down version of the static recording environment is shown in figure 5.2. It is intended to run full screen on the recording monitor. It consists of a single large image, two on-screen buttons, an instructional text, and a tracking status indicator. One button enables and disables the eye-tracking data stream, after which a heat map and a time-series data file are generated. A second button determines how outgoing data samples are pre-labeled. Possible labels are fixation, saccade, and blink. The button is interactable to indicate when the user is fixating somewhere on the image or blinking. On-screen stimulus is in the form of a highly abstract and visually stimulating image. It is chosen so no area of particular interest draws the user's attention since that might introduce unwanted bias in the data.

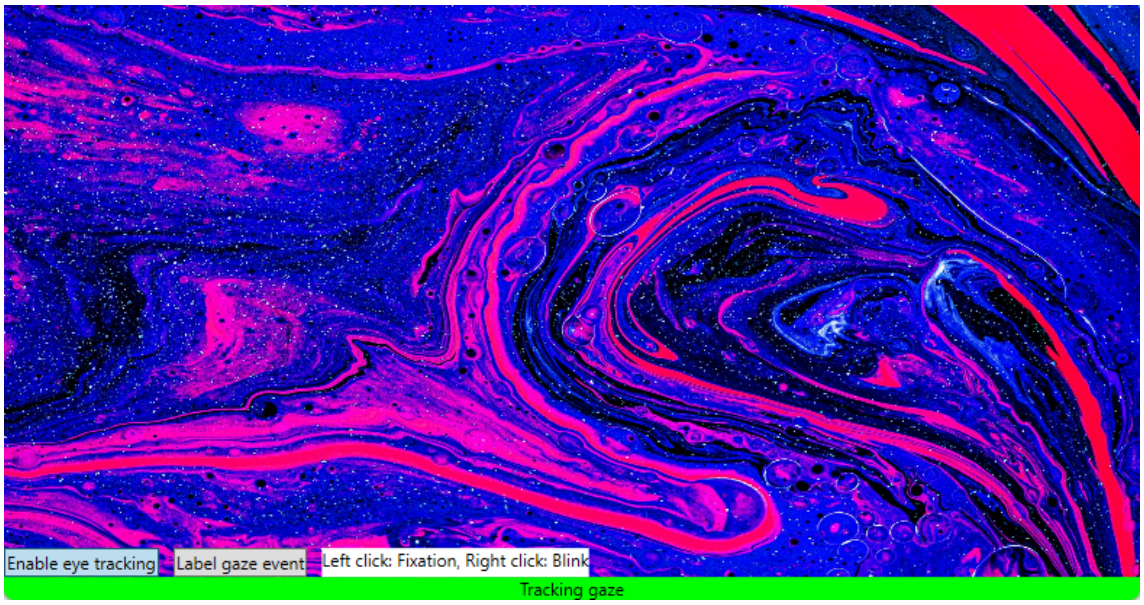

Figure 5.2: Static labelling environment, used to label data for the binary classification problem.

5.2.2 Dynamic

The dynamic recording environment improves upon the above by replacing static stimulus with a graphical element able to animate the movement of a white dot on a black screen. Movements are random within some constraints and include instantaneous warping and slow displacements. Displacements are along straight lines or second-degree parabolas and between two and five seconds in duration. The dot rests for at least two seconds between displacements. For every rest and movement of the stimulus, outgoing data samples are pre-labeled with either fixation, saccade, or smooth pursuit. An example snapshot of the environment, where the moving dot leaves a path along with its movement, is shown in figure 5.3. This figure also depicts the heat map that came from that particular recording. A heat map and a time-series data file are generated upon exiting the application.

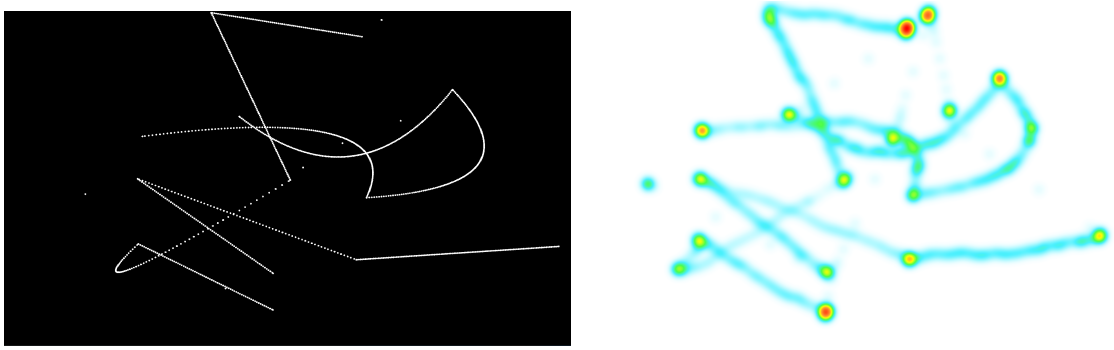

Figure 5.3: Dynamic Labelling environment. Left image represents the paths drawn by warping and moving dot during data recording. Right image is the heat map resulting from the same recording. Note that this example is artifially created to draw lines along displacement paths. During normal operation, there is only one dot on-screen at any time.

5.3 Labeling

For reasons explained in section 2.2, supervised learning requires a labeled dataset to train the learning algorithm for classification. For the specific classification problem which this thesis presents, a dataset with labeled eye movement events is required. These labels will constitute

the ground truth for model training. Therefore, its quality will directly translate to the quality of the model produced and the accuracy of predictions. Since the quality of a labeled dataset is tied to the labeling method, this is a crucial step in training a classification algorithm.

As mentioned in the section above, pre-labeling is done while data is recorded. Even so, these labels are likely to be inconsistent with reality, as they make some assumptions on human reaction times that are not necessarily accurate. Therefore, a second step in the data acquisition pipeline is added to ensure the quality of labeled eye movement events. Here, an operator scans the pre-labeled data files output from the previous step. A .csv editor and a simple plotting script implemented in Python simplify the process. For each event that is not pre-labeled as a fixation, the operator then modifies labels sample-by-sample to match reality.

5.4 Feature Extraction

The final step of the data acquisition pipeline is responsible for preparing the dataset before classification. Here, if blinks are not already labeled from the static environment, they are automatically inferred by the time difference between samples. Such samples are considered data contamination and removed.

Finally, features are appended to the dataset, which the classifier will use to distinguish between events. Since the machine learning model to be detailed in the next section is rather limited in its hypothesis space, it requires more than just coordinate values to do accurate classifications. This alternate representation is acquired by extracting relevant information from the dataset. The choice of features is inspired from Zemblys, Niehorster et al. 2018, which is again inspired by common or state-of-the-art manual classification algorithms. However, since the Tobii ET5 has a significantly lower sampling rate than the eye tracker used in this paper, many features that rely on small sample intervals have been eliminated. The eight features extracted are presented in table 5.1.

Feature	Description
rms	*Root mean square* of the coordinate values in a 150ms sample window (5 samples at 33Hz) centered on the sample. Calculated individually for both x- and y-axis.
std	*Standard deviation* of the of all coordinate values in a 150ms sample window (5 samples at 33Hz) centered on the sample. Calculated individually for both x- and y-axis
disp	*Dispersion* of all coordinate values in a 150ms sample window (5 samples at 33Hz) centered on the sample. This is the same property that is used to determine fixations in the IDT-algorithm detailed in section 3.2. Calculated as ((max(x)) - min(x)) + (max(y) - min(y)))
vel	*Velocity* from previous sample in time. Note that there is little risk of "exploding" velocities from the derivation of noise since the output from Tobii ET5 is sufficiently filtered. This is the same property that is used to determine saccades in the IVT-algorithm detailed in section 3.2. Calculated as the euclidean distance between points times the sampling frequency.
med-diff	*Difference* between median value of 150ms sample window before the sample and an equally sized window after the sample. Calculated individually for both x- and y-axis.
mean-diff	*Difference* between mean value of 150ms sample window before the sample and an equally sized window after the sample. Calculated individually for both x- and y-axis.
rms-diff	*Difference* between root-mean-square of 150ms sample window before the sample and an equally sized window after the sample. Calculated individually for both x- and y-axis.
std-diff	*Difference* between standard deviation of 150ms sample window before the sample and an equally sized window after the sample. Calculated individually for both x- and y-axis.

Table 5.1: Features extracted from dataset.

5.5 Classification

Once eye movement data has been passed through the data acquisition pipeline, the classification dataset is imported to a Jupyter Notebook. The classification itself is done by three distinct models, implemented in object-oriented `Python`.

One of the three models is a Random Forest Classifier, the background of which was explained in section 2.2. This particular model was chosen for its proven dominance in eye movement event classification when compared to nine other machine-learning algorithms (Zemblys 2016). It was implemented using the `Python` package `Scikit-Learn` (Pedregosa et al. 2011), which allows for a streamlined training process without excessive overhead. Besides the default parameters, information gain was used to measure the quality of a split, and trees were given no maximum depth. Additionally, the weight of each class was set to be inversely proportional to class frequencies in the data. This was done to account for a surplus of smooth pursuit and fixation classes compared to saccades. Model training was done on a dataset generated by about five minutes of recorded eye movement data in the dynamic recording environment, with the author as the subject.

The second and third models implemented one of each of the two algorithms defined in section 3.2. Note that sample-to-sample velocities were already calculated during the feature extraction step. As such, algorithm 2 (IVT) was simplified by the removal of lines 2-3. Also note that this is not the case for the IDT algorithm, as it requires the calculation of dispersion for a variable-sized sample window. Since the IVT algorithm only classifies saccades and the IDT algorithm only fixations, all other samples were set to be undefined.

When the first model was fully trained, two final datasets were made to compare the models. Since the IDT- and IVT algorithms are not designed to be used on datasets with smooth pursuit, one dataset was generated in the dynamic recording environment by excluding all stimulus movement except warping. The other dataset was generated with the same settings of the recording environment with which the training dataset was recorded.

Chapter 6

Results

6.1 Data Quality

The results presented will paint a picture of the quality of data produced by the eye tracker detailed in section 4.2. All eye movement data was generated on a 24" 1920x1200 (16:10) Dell monitor.

6.1.1 Qualification Tests

Results from the first qualification test conducted are presented in figure 6.1 and aim to provide a general overview of data quality. Results were generated by instructing the subject to fixate on nine different points for three seconds each. These nine points were chosen to observe how gaze is output at differing degrees of visual angle, as well as edge cases. The farthest points were at the four very edges of the monitor. One was at the center and the rest on the diagonals in between. Around 100 samples were collected for each gaze point. The test was subsequently repeated three times in a controlled environment. That is, with unchanged device calibration while disallowing the subject to move in the tracking space.

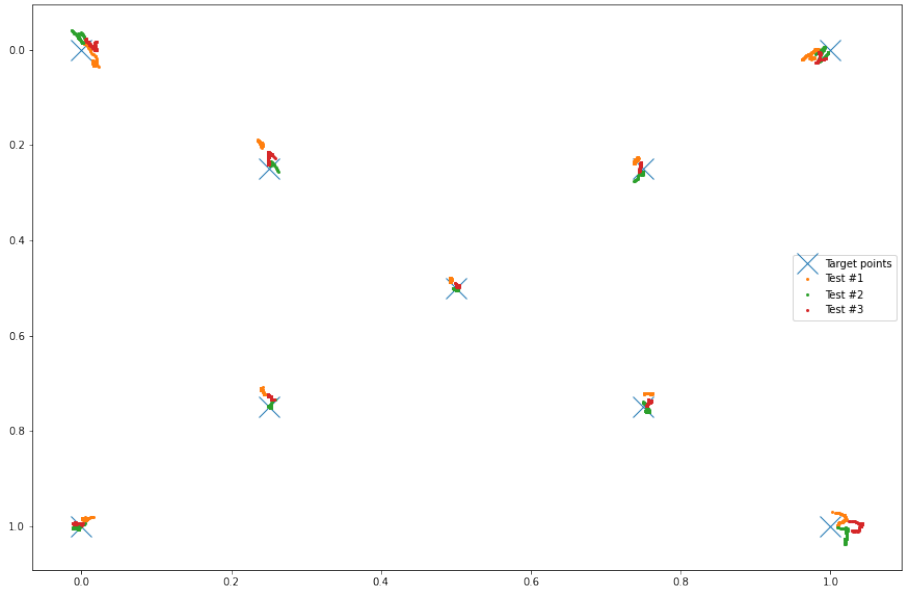

Figure 6.1: Gaze point test results. The x- and y-axes represent on-screen coordinates, as they are output from Tobii ET5. The blue crosses are the gaze points on which the user was instructed to fix their gaze. Orange, green, and red dots represent single-sample representations of user gaze in the first, second, and third tests, respectively.

Another qualification test was conducted to see whether scan paths produced by Tobii ET5 were consistent. Results are presented in figure 6.2. This time, the subject was instructed to follow a moving target, traveling smoothly along all four borders of the monitor and along both diagonals. Again, the test was repeated three times. Each test lasted about 30 seconds, and around 3000 samples were collected in all.

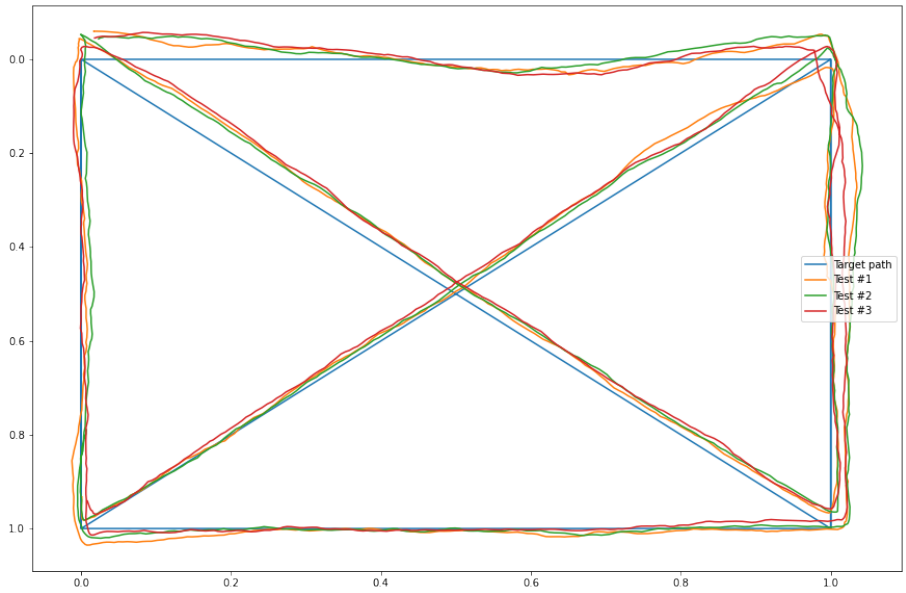

Figure 6.2: Scan path test results. Once again, the x- and y-axes represent on-screen coordinates, as they are output from Tobii ET5. The straight blue lines are the target paths which the user was instructed to follow with their gaze.

Behav Res (2018) 50:160–181
DOI 10.3758/s13428-017-0860-3

Using machine learning to detect events in eye-tracking data

Raimondas Zemblys[1,2] · Diederick C. Niehorster[3,4] · Oleg Komogortsev[5] · Kenneth Holmqvist[2,6]

Published online: 23 February 2017
© Psychonomic Society, Inc. 2017

Abstract Event detection is a challenging stage in eye movement data analysis. A major drawback of current event detection methods is that parameters have to be adjusted based on eye movement data quality. Here we show that a fully automated classification of raw gaze samples as belonging to fixations, saccades, or other oculomotor events can be achieved using a machine-learning approach. Any already manually or algorithmically detected events can be used to train a classifier to produce similar classification of other data without the need for a user to set parameters. In this study, we explore the application of random forest machine-learning technique for the detection of fixations, saccades, and post-saccadic oscillations (PSOs). In an effort to show practical utility of the proposed method to the applications that employ eye movement classification algorithms, we provide an example where the method is employed in an eye movement-driven biometric application. We conclude that machine-learning techniques lead to superior detection compared to current state-of-the-art event detection algorithms and can reach the performance of manual coding.

Keywords Eye movements · Event detection · Machine learning · Fixations · Saccades

Introduction

In eye movement research, the goal of *event detection* is to robustly extract events, such as fixations and saccades, from the stream of raw data samples from an eye tracker, based on a set of basic rules and criteria which are appropriate for the recorded signal. Until recently, researchers who ventured to record eye movements were required to conduct time-consuming manual event detection. For instance, Hartridge and Thomson (1948) devised a method to analyze eye movements at a rate of 10001 s (almost 3 h) of analysis time for 1 s of recorded data, and as Monty (1975) remarked: "It is not uncommon to spend days processing data that took only minutes to collect" (p. 331–332).

Computers have fundamentally changed how eye movement data are analyzed. Today, event detection is almost exclusively done by applying a detection algorithm to the raw gaze data. For a long time, two broad classes of algorithms were used: First, the *velocity-based algorithms* that detect saccades and assume the rest to be fixations. The most well-known is the I-VT algorithm of Bahill, Brockenbrough, and Troost (1981) and Salvucci and Goldberg (2000), but the principle can be traced back to algorithms by Boyce from 1965 and 1967 as referred to by Ditchburn (1973). The *dispersion-based algorithms* instead detect fixations and assume the rest to be saccades. The best known is the I-DT algorithm of Salvucci and Goldberg (2000).

✉ Raimondas Zemblys
r.zemblys@tf.su.lt

1 Department of Engineering, Siauliai University, Siauliai, Lithuania
2 Humanities Laboratory, Lund University, Lund, Sweden
3 Humanities Laboratory and Department of Psychology, Lund University, Lund, Sweden
4 Institute for Psychology, University of Muenster, Muenster, Germany
5 Department of Computer Science, Texas State University, San Marcos, TX, USA
6 UPSET, NWU Vaal, Vanderbijlpark, South Africa

Springer

Figure 6.3: Heat map test. Color gradient from cyan to red indicate fixation duration from brief to long. The right figure shows the paper that was read.

Figure 6.3 attempts to illustrate a possible application of eye-tracking data with one final test. Here, the subject was instructed to read one page of a paper and terminate the recording once the task was complete. In the end, a heatmap was generated from the data. Both heatmap and paper are presented here.

6.1.2 Other Insights

Fundamental statistical properties of the data stream are given in figure 6.4. These violin plots were all generated on the data gathered during the first repetition of the gaze point test of figure 6.1. Each violin of each plot represents the deviations in on-screen coordinates on one axis from the mean of one observed set of data samples. Since the mean values of each point are subtracted from each violin, these plots do not account for bias. Each of these sets of data samples comes from the user fixating at one of the nine gaze points of figure 6.1. Plots in the same column are generated from the same target gaze point groups. Groups are organized in edge, diagonal, and center points. The top row of plots shows deviations in the y-coordinate axis, while the bottom row shows the same on the x-axis.

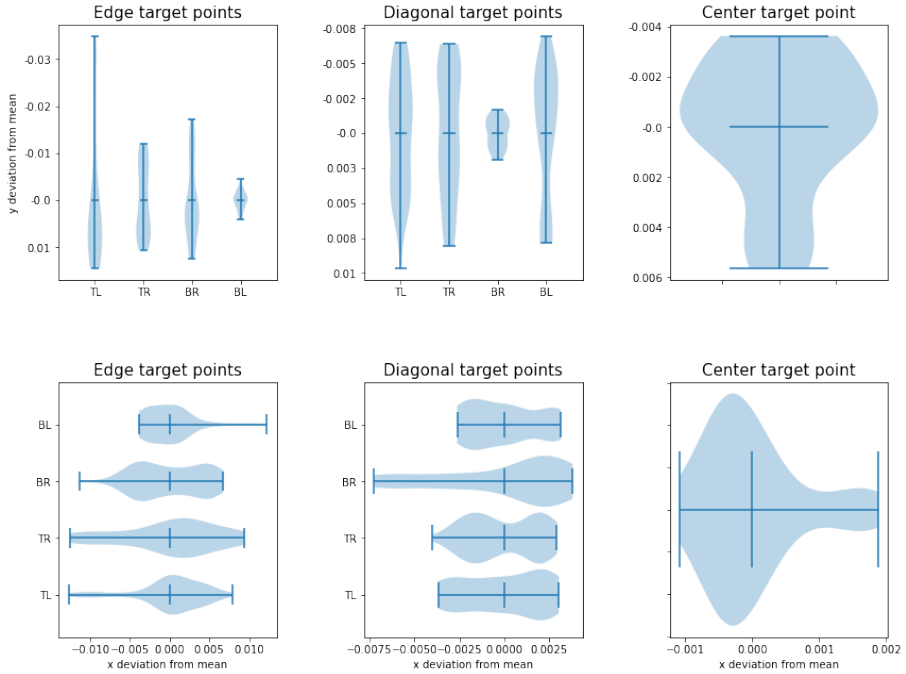

Figure 6.4: Data deviations. Columns share target gaze point types, rows share coordinate axes. For edge- and diagonal target plots, each violin represent either the top-left (TL), top-right (TR), bottom-right (BR), or bottom-left (BL) gaze target points. Note that the coordinate axes are scaled to match the values of each plot.

Accompanying figure 6.4 is table 6.1, which illustrates typical occurrences of bias and variance in the data output. As with figure 6.4, this data was gathered from the first iteration of the gaze point test, representing related information. Additionally, bias is inferred by individually calculating the median position on both coordinate axes and subtracting the target point coordinates.

Target point		**C**	**TL**	**TR**	**BR**	**BL**
Center	**Std**	(0.001, 0.003)	-	-	-	-
	Bias	(-0.007, -0.018)	-	-	-	-
Diagonal	**Std**	-	(0.002, 0.004)	(0.002, 0.004)	(0.003, 0.001)	(0.002, 0.005)
	Bias	-	(-0.01, -0.055)	(-0.01, -0.019)	(0.01, -0.029)	(-0.009, -0.036)
Edge	**Std**	-	(0.004, 0.014)	(0.006, 0.007)	(0.004, 0.008)	(0.003, 0.002)
	Bias	-	(0.016, 0.025)	(-0.025, 0.011)	(0.014, -0.01)	(0.004, -0.016)

Table 6.1: Statistical measures for data quality. Each entry represent the standard deviation or bias on both coordinate axes as (X, Y).

Finally, the effects of data drift are shown in figure 6.5. This plot is taken from all repetitions of the gaze point test of figure 6.1 and is merely centered on and scaled up to only show the fine-grained evolution of gaze points during fixation. This particular point is at the top-left corner of the monitor.

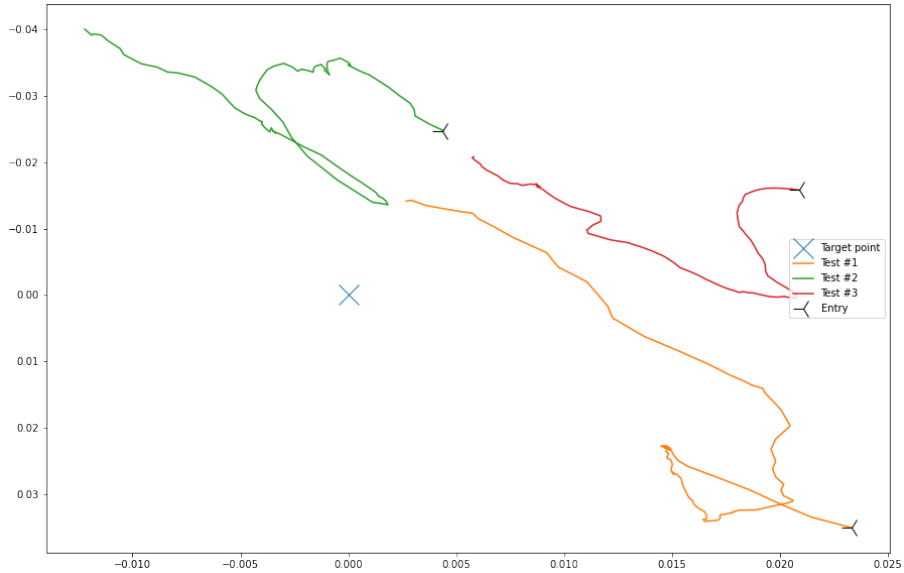

Figure 6.5: Effects of data drift. Again, the x- and y-axes represent on-screen coordinates, as they are output from Tobii ET5. The single blue cross is the top-left target gaze point observable in figure 6.1. Orange, green, and red lines show the between-sample scan paths of user gaze in the first, second, and third tests, respectively. The black tri-crosses are the entry points from which the scan path evolves.

6.2 Datasets

6.2.1 Environment Pre-labeling

Figure 6.6 attempts to visualize the quality of datasets and labels before the post-labeling process in the data acquisition pipeline is applied. Plots are generated from a small, random subset of samples. They are grouped such that both x- and y-coordinates of gaze can be observed parallel to the XY-dispersion feature, discussed in section 5.4. Since this feature likely will be one of the primary variables to distinguish classes during classification, it serves as a good indicator of the accuracy of labels against the ground truth.

6.2.2 Final dataset

Finally, after data has been recorded, imported, processed, and post-labeled, we get the dataset from which the subset presented in figure 6.7 is taken. By comparing this plot to the right one of figure 6.6, one can observe that they represent the same set of eye-tracking data samples. This dataset is the one that will be utilized to train the final classification model.

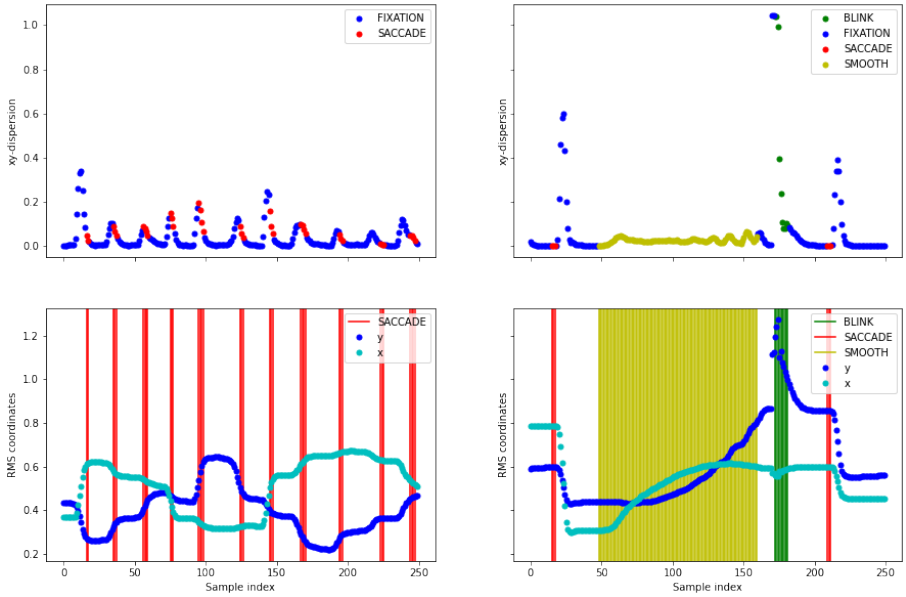

Figure 6.6: Dataset as output from static (left) and dynamic (right) recording environments. All plots show individual samples as dots, plotted against the RMS-value of their raw coordinates (bottom) and XY-dispersion (top). Red, green, and yellow samples are color-coded such that dots (top) and vertical lines (bottom) signify samples that are labeled saccades, blinks, and smooth pursuits, respectively. To avoid cluttering, fixations are blue in the top plot and white in the bottom.

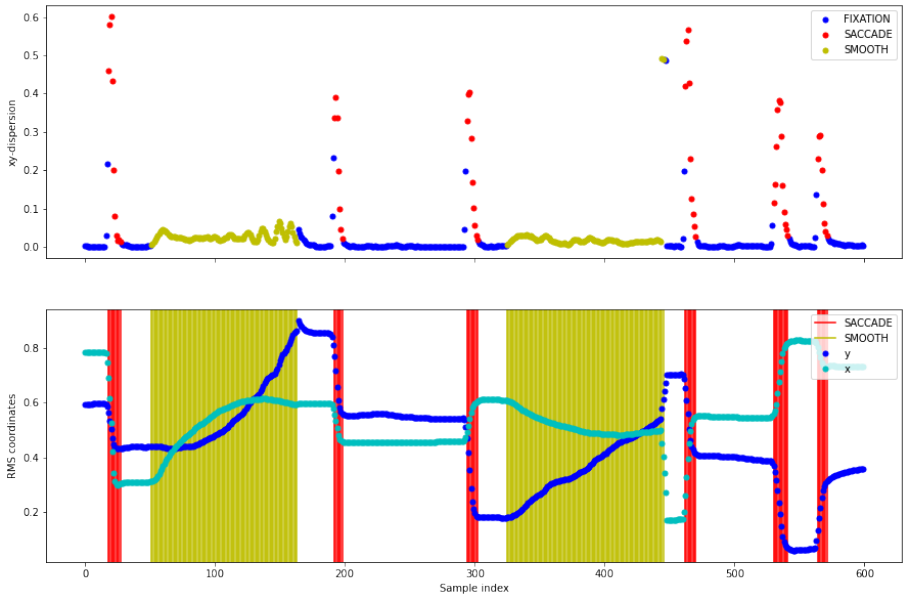

Figure 6.7: Final dataset to be used for classification. Axes and labels are identical to that of figure 6.6.

6.3 Classification

6.3.1 Random Forest Classifier Feature Importances

One interesting property that is inherent in Random Forest Classifiers is illustrated in figure 6.8. Here, each bar shows the relative importance of all features discussed in section 5.4. In other words, the features with high relative importance have a high significance when the class of a sample is to be determined.

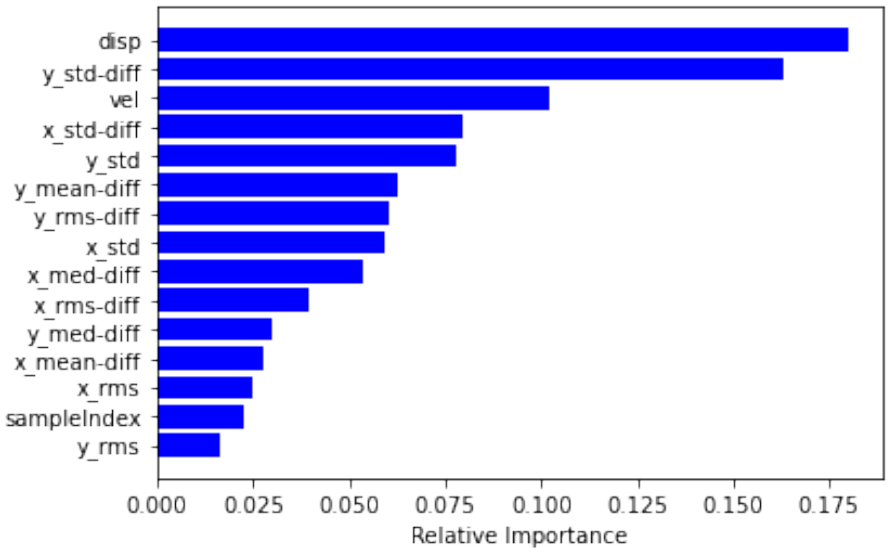

Figure 6.8: Feature Importance. Each bar show the relative importance of every feature present in the machine learning model. For every bar, the y-axis display the features they represent.

6.3.2 Algorithm Comparisons

Figures 6.9 and 6.10 were both generated by running the same dataset through three classification algorithms. For figure 6.9, the dataset was binarily labeled to illustrate the applicability of manually coded algorithms. For figure 6.10, however, the inherent strength of the proposed machine learning model was shown by utilizing a dataset including smooth pursuit. The acquisition of these datasets was detailed in section 5.5.

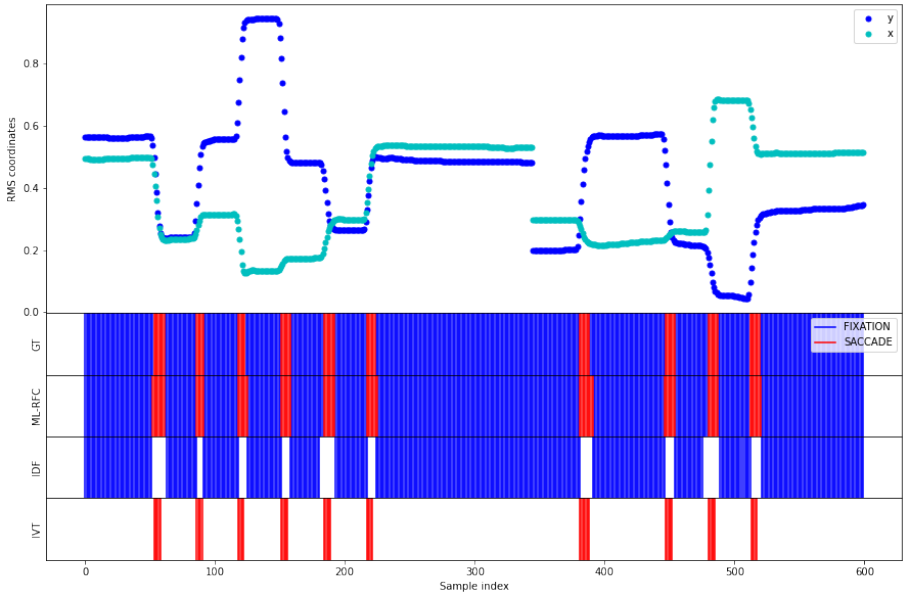

Figure 6.9: Binary classification results. The top plot represent sample-by-sample RMS-value of on-screen x- and y-coordinates as cyan and blue dots, respectively. The next four rows represent (from top to bottom) the ground truth (GT), ML-model classifications (ML-RFC), IDF-algorithm classifications (IDF), and IVT-algorithm classifications (IVT). Blue segments correspond to fixations and red correspond to saccades.

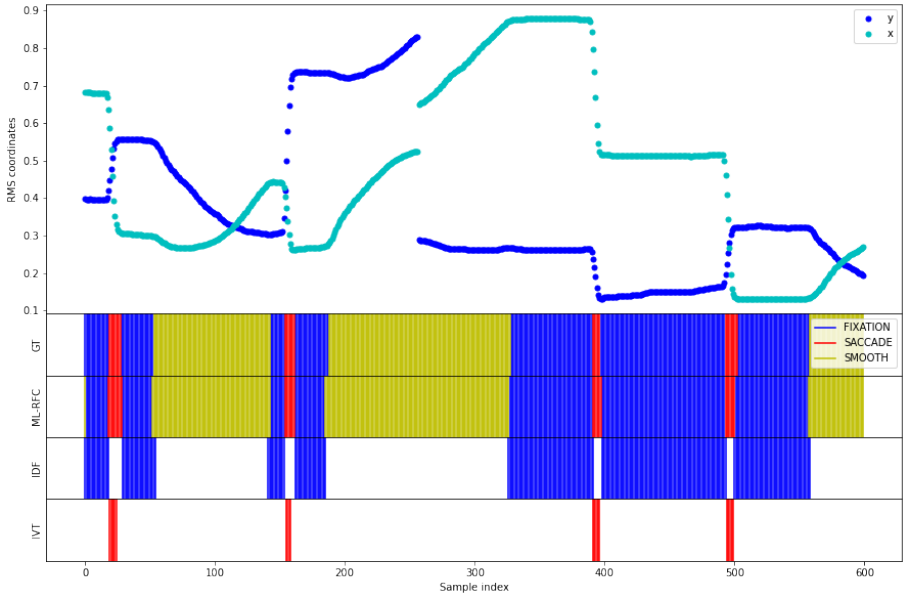

Figure 6.10: Multi-class classification results. See figure 6.9 for description.

Chapter 7

Discussion

7.1 Tobii ET5 Evaluation

From the data quality results of section 6.1, we get a broad overview of the performance of Tobii ET5. The data quality is good, abundant, and relatively frequent. However, as is immediately apparent from both figures 6.1 and 6.2, there is a significant effect of bias, and some points on-screen are more disturbed than others. Corner points, particularly top corner points, seem to consistently output values that are slightly off their target values. This result is reasonable, considering how it calculates on-screen gaze points and where the tracker is mounted. By the horizontal fashion in which the IR- and camera sensors are placed in the tracker's casing, it is not unnatural that noise and measurement error is more pronounced for the detection of vertical eye movements. Additionally, since the tracker's mounting point is right below the center of the monitor, measurement error becomes more and more apparent the farther one is looking from this point

In terms of data consistency, the results from 6.2 are very interesting. From this plot, we can see that although the scan paths tracked by Tobii ET5 are slightly off their target, they are remarkably consistent between consecutive tests. This consistency shows that the bias observed in both the gaze point and scan path tests might primarily be caused by poor tracker calibration or difficulties in designing a system that can discern gaze points universally between a wide variety of monitors and recording environments. Another interesting result from the scan path test is the peculiar dip in y-position when passing along the top border of the monitor. One explanation for this effect might be that the internal processing within the tracker recognizes that it consistently outputs values outside of the defined [0.0, 1.0] range and attempts to correct accordingly. Another and perhaps more likely explanation is that the sensor has trouble discerning purely lateral movements and slightly tilted lateral movements when the gaze is on the very ends of the screen, where it already struggles the most with calculation. Conversely, the scan path is very accurate on the opposite end of the screen, right above where the tracker is mounted.

The variance in data output is visualized in figure 6.4 and given with hard numbers in table 6.1. However, since these results are based on just one empirical test, and we have established that the Tobii ET5 output rather unstable data, they should be taken with a grain of salt. From the violins of each plot, we can see deviations in coordinates from the mean value of one point on one axis. What is especially interesting here reflects what was discussed above: the fading data quality at the top and edge target points compared to the center point. The effect is also particularly noticeable on the y-axis, with generally higher deviations on the top row of violin plots, as well as the absolute value of the second std-entry in the coordinate tuples of 6.1. The same consequence is seen in the bias tuples, with higher values on the y-axis entry seven out of nine times. The bias entries also almost consistently tend upwards, indicated by negative values on the y-axis. This evidence and the fact that all variance entries lie within relatively low magnitudes argue for the point made above. Therefore, we can confidently conclude that the data output from Tobii ET5, although fairly biased towards higher vertical points, is very accurate.

Another thing to note is the shape of the violins of figure 6.4, with either long tails in one direction or something resembling two Gaussian distributions with a "dip" between them. If the data were only affected by bias and variance, one would expect to see only one Gaussian distribution centered on the mean value of all samples (or the target point if there were no biases).

However, the results we see here are likely caused by data drift. The shape produced depends on whether the signal continuously drifts in one direction during recording or if it at some point "turns" and drifts in the opposite direction. If we take a closer look at the data gathered by the three-second fixation tests at the top-left target of figure 6.1, we can observe the very erratic gaze behavior of figure 6.5. From these results, it is clear that drift is significant. It seems that there is little noise in the data at all and that the data deviations discussed above might be entirely caused by data drift. As detailed in section 4.2, however, the "raw" data we get from Tobii ET5 is in reality lightly filtered for stability.

Finally, we see an actual use case for the eye tracker in figure 6.3. From this plot, none of the shortcomings discussed above are truly noticeable except perhaps some rightward bias in the lower regions of the screen, causing distortion. Nonetheless, one can clearly distinguish fixations on paragraphs from one another and even individual lines of text in some cases. The value of this data is self-evident. From this simple test, one can infer that the subject has paid particular attention to the abstract paragraph while seemingly glancing over other sections. They also seem to have completely ignored the header and foot-notes, and of the authors, they were the most focused on capturing the primary author.

7.2 Recording and Labeling

The development of recording environments has been an iterative process. It begins at the barest minimum viable product and ends with an entirely plug-and-play robust application. In the end, it was able to handle all aspects of eye-tracking data management, from data recording to heatmap generation and pre-labeling. Since the raw implementation and operation of such environments were described in section 5.2, this section will focus on describing the datasets and labels they were able to produce, as well as the reasoning behind the most important choices made along its developmental process.

7.2.1 From Static to Dynamic

To begin with, the goal of the recording environment was simple: To connect to hardware, gather raw eye-tracking data, and output a dataset sufficient for binary classification. Since this requires no more than binary labels, the entire labeling process could essentially be done at runtime by the subject. For this purpose, a simple environment with a static image and an on-screen labeling button would suffice. The thought was that the user could hold a button when fixating on a point on the image and release it when the focus shifted. Additionally, the user could indicate that blinks contaminated the data output by pressing another button to stop recording temporarily.

As the reader might already have reasoned, this labeling scheme is sub-optimal. To obtain a dataset of high quality with no external biases, we ideally want the subject to take no part in the recording process itself and only focus on the task at hand. Furthermore, if we wanted to use the dataset for more than binary classification, the burden of labeling more than two eye movement events during runtime would be too complex for the subject to handle.

For these reasons, a natural augmentation was that of a dynamic environment, the details of which are explained in section 5.2. This development removed the subject from the loop entirely, allowing for less biased and more abundant data since they could record for extended periods before getting tired. Another unexpected improvement of the dynamic environment was the effect of discovering and removing more contaminations by blinks. For the static environment, it turned out that even though the subject was instructed to label their blinks, there were inevitably unconscious blinking that went unnoticed in the final dataset. Even when run through post-labeling, a manual agent found these minor data points, distinguishable only by a few milliseconds larger gaps between timestamps of consecutive samples, very hard to notice.

The raw labeled datasets as output from both the static and the dynamic recording environments are depicted in figure 6.6.

7.2.2 The Value of Post-labeling

Observing the onsets and offsets of movement events in both datasets of figure 6.6, however, there is still a very poor correlation between labels and the ground truth. For the statically recorded dataset, this is again caused by involving human error in the labeling process. Both because human reaction time is unpredictable and nonzero and because the user has no way of distinguishing differing lengths of movements in their labeling, the resulting labels are lagging and randomly distributed in their quantity. The dynamically recorded dataset is hardly any better, however. It, too, suffers from error caused by human reaction time, though in the opposite direction in time. Another likely cause is that the data stream from Tobii ET5 is also lagging actual events by some milliseconds. Either way, to obtain the dataset we seek for accurate event classification, these errors need to be addressed.

The solution manifests itself as a manual post-labeling step in the data acquisition pipeline. Although sub-optimal regarding the autonomy of the labeling process, some kind of manual intervention is likely unavoidable to obtain a dataset with labels consistent with reality. Still, knowing that the events labeled by the environment are actual and merely shifted a few samples behind the ground truth, post-labeling becomes a straightforward and smooth process. The final dataset, post-labeled and with contaminated blink samples removed, can be seen in figure 6.7. As mentioned, this particular subset of samples includes those from the right plot of figure 6.6. From this, one can ascertain the value of post-processing by observing that labels have been both shifted and stretched in time to resemble the ground truth as closely as possible. Additionally, one can see that most data contaminated by subject blinking is removed.

7.3 Classification

7.3.1 Binary

Section 3.2 served as an introduction to some of the manually coded algorithms available for eye event classification. The author focused on the IDT and IVT algorithms for this thesis, mainly for their simple implementations. Another factor, however, was that most newer and more advanced alternatives are heavily reliant on an estimate of eye movement velocity. Since this metric is limited for eye trackers with relatively low-frequency sampling rates, it would prove challenging to fully implement on our low-cost commercial hardware.

The performance of these algorithms is visualized in figure 6.9. However, for reasons elaborated in section 3.2, they do not explicitly classify both fixations and saccades but instead assume the unclassified event to be the other. For this reason, a characteristic of most manually coded eye event classification algorithms is that they require data utterly void of anything but saccades and fixation to ensure proper function. However, for the binary classification problem and with this fact in mind, they perform very well, as is evident by their near-perfect correlation with the ground truth.

7.3.2 Multi-class

It is upon observing figure 6.10 that the limitations of manually coded algorithms truly become apparent. Here the dataset to be classified includes fixations, saccades, and smooth pursuit. Indeed, as mentioned above, these algorithms immediately struggle to fully classify events that correspond with the ground truth when more than two classes are introduced. Since all events that are not fixations (for IDT) or saccades (for IVT) are assumed to be either saccades (for IDT) or fixations (for IVT), smooth pursuits are left unclassified. The Random Forest Classifier model, however, solves this problem impressively well. Since it makes classifications based upon several distinct features of the dataset, it is able to distinguish between all three classes.

What seems particularly interesting from the classification results of figure 6.10 is that if one were to implement both the IDT and the IVT algorithm in sequence, the presence of smooth pursuits could be recognized as their collective unclassified samples. In fact, this is likely exactly what the machine learning model does under the hood. From figure 6.8, it is evident that the dispersion and velocity features detailed in section 5.4 play a significant role when the distinction between classes is to be determined. Recalling from 3.2, these are the same features on which the IDT and IVT algorithms determine classification, so the above result is not surprising.

From the figures discussed, it seems clear that our machine learning model emerges as the most accurate and robust classification method when multiple classes need to be determined. However, since it requires a reasonably large and well-labeled dataset to train, it is not necessarily the best when only binary classification is needed. The manual operation of parameter tuning is usually considered one of the principal drawbacks of manually coded algorithms. Despite this, experience shows that the tuning of a machine learning model is undoubtedly a drawback of its own. Therefore, manually coded algorithms are likely still optimal for simple classification problems where only one feature is required for accurate classifications.

7.3.3 Model Performance and Improvements

Results are surprisingly accurate, suggesting that low-cost eye-trackers do not necessarily perform much worse than high-frequency eye-trackers. The limitation is mainly on the choice of features, as high-fidelity metrics are unavailable when the recording frequency is low.

While the Random Forest Classifier solves the initial eye event classification problem, it is not even close to the most optimal. Sadly, the author did not research the prospects enough to implement such a model. Some solutions come to mind, however. For instance, one particular shortcoming of using a classical machine learning model is the need for manually designed features. As described in section 5.4, all features used for classification are selected by an educated assumption of its inherent correlation with eye movement. However, an educated assumption is still merely an assumption, so the model would likely significantly improve if it generated its features as a set of hidden layers in a neural network.

Chapter 8

Closing Arguments

8.1 Further Work

8.1.1 Model Improvements

Recurrent Neural Networks

As discussed above, our method mainly focused on classical machine learning to perform eye movement classification. However, considering the sequential nature of eye movement data, a potentially significantly improved classification model may instead be in the form of a Recurrent Neural Network. For reasons elaborated in section 2.2, this form of deep learning has shown impressively accurate classifications on data sets where there is a nonzero statistical relationship between present and past sample labels. First, this method will eliminate an inherent disadvantage of classical machine learning: the necessity of manually defining feature sets before classification. Second, it is reasonable to believe that we might see results more in tune with reality, avoiding the need for post-processing. For this reason, further work should consider implementing such neural networks in place of the classification models discussed here to solve a similar classification problem.

Blink Imputation

Another advancement that might significantly improve classification performance is imputation techniques such as Multiple Imputation by Chained Equations (MICE) on missing data caused by user blinks, rather than simply ignoring such data points. The labels of imputed samples would match that of either the preceding or the following sample. Of course, doing so would impose a source of uncertainty proportional to the amount of data imputed. As such, one could introduce a hyperparameter that only performs the process on missing data windows smaller than a given size. On the other hand, imputation would significantly improve the data set by incorporating data that would otherwise be considered contamination and removed. Considering the blink rate of an average user, this makes for a significant amount of potentially valuable information that would improve model training.

8.1.2 Advanced Eye Data Inference

By now, we have firmly established that one need not have the most expensive research-grade hardware to make basic inferences on eye-tracking data, but need this be the end? With more advanced classification models and new data labeling schemes, there is likely potential in other analyses besides simple eye movement classification. Additionally, as was briefly mentioned in section 4.2, there are a lot of data streams readily available by the Tobii eye-tracker which were not explored in the methods of this thesis. Of particular interest is head pose and distance from the display, individual left- and right-eye gaze points, and pupil diameters.

There is no lack of scientific literature linking various ocular activities with cognitive states. Following are some exciting correlations, to name a few. First and foremost, there is a strong consensus that a user's field of view and pupillometry is closely related to cognitive processes and mental workload (Kahneman and Beatty 1966, May et al. 1990, Chen and Epps 2014, Seeber

2013). In fact, the onset, offset, and magnitude of pupillary dilation likely corresponds to the onset, offset, and difficulty of mental tasks (Beatty 1982). Additionally, some researchers believe that blink rate too is related to the same kind of load, as well as possibly sleep deprivation and even brain dopamine levels (Barbato et al. 1995). Although the literature is promising, the brain is an unimaginably complex organ with states and processes which we can never fully understand in hard numbers and interconnections.

Yet, with large enough data sets and machine learning models that encapsulate a sufficiently large hypothesis space, there is reason to believe that there is a real possibility of inferring information about a person's mental state by analyzing eye-tracking data.

8.2 Conclusion

Through this thesis, the focal point has persistently converged towards the commercial availability of eye-tracking. The reason is rooted in a paradigm shift that will drive eye trackers from research labs and into the homes of ordinary people within just a few years. To echo the introduction, this calls for new applications where even the cheapest eye tracker is applicable, and to that end, this thesis merely does the groundwork.

Section 7.1 concluded that the eye tracker that was used here could output data that is remarkably accurate, although sensitive to drift. However, what was surprising was the impressive consistency in data when gaze was not fixed for extended periods. This result notably speaks for the application of eye-tracking in gaming, where the user rarely fixes their gaze for long periods at a time. Consistency is also vital for machine learning applications, as a trained model would hardly be generalizable for production if different eye trackers output conflicting data. As such, results show that eye trackers need not cost tens of thousands of dollars to present a value to the user, and a cheap commercial eye tracker is often enough to do most kinds of statistical inferences.

This result was made possible by implementing a recording environment, which proved to be vital in labeling datasets to establish a ground truth by which to both train and validate classification methods. Although the author did not manage to automate the process entirely, as was the aspiration, the resulting application streamlined the labeling process significantly compared to manual methods.

With the datasets from this application, the author proposed a classification method using a straightforward machine learning model which was readily able to compete with established methods of manually coded algorithms. This thesis did not convene on a specially designed model that would perform beyond a simple benchmark. However, as was discussed, there are many unexplored potentials in this regard. For instance, implementing a recurrent neural network could infer relations between time samples as a trainable parameter. Doing so could remove work from the designer and reveal features that would otherwise not have been readily understandable by human intuition.

Eye-tracking is likely a field of technology that will accelerate tremendously in the coming years. This development might, in time, make such hardware as common as the keyboard or mouse. If this happens, it is astonishing what insights can be made on the user's account. And it is not limited by eye movement classification. The horizons stretch beyond automated personal coaching, eSports talent scouting, feedback on mental state, and more. As long as there is data, correlation, and a neural network that encompasses a complex enough hypothesis space, anything is possible.

References

Antunes, João and Pedro F. Santana (2018). 'A study on the use of eye tracking to adapt gameplay and procedural content generation in first-person shooter games'. In: *ArXiv*.

Ballard, Dana H. et al. (1992). 'Hand-Eye Coordination during Sequential Tasks [and Discussion]'. In: *Philosophical Transactions: Biological Sciences* 337.1281, pp. 331–339.

Barbato, Giuseppe et al. (1995). 'Effects of sleep deprivation on spontaneous eye blink rate and alpha EEG power'. In: *Biological Psychiatry* 38, pp. 340–341.

Barry, Philip et al. (1994). 'Intelligent Assistive Technologies'. In: *Presence: Teleoperators and Virtual Environments*.

Beatty, Jackson (1982). 'Task-evoked pupillary responses, processing load, and the structure of processing resources.' In: *Psychological bulletin* 91 2, pp. 276–92.

Castelhano, Monica and Keith Rayner (Jan. 2008). 'Eye movements during reading, visual search, and scene perception: An overview'. In.

Chen, Siyuan and Julien Epps (Apr. 2014). 'Using Task-Induced Pupil Diameter and Blink Rate to Infer Cognitive Load'. In: *Human-Computer Interaction* 29.

Corno, F., L. Farinetti and I. Signorile (2002). 'A cost-effective solution for eye-gaze assistive technology'. In: *Proceedings. IEEE International Conference on Multimedia and Expo*.

Eckstein, Maria K. et al. (2017). 'Beyond eye gaze: What else can eyetracking reveal about cognition and cognitive development?' In: *Developmental Cognitive Neuroscience* 25. Sensitive periods across development, pp. 69–91.

Gog, Tamara van and Halszka Jarodzka (2013). 'Eye Tracking as a Tool to Study and Enhance Cognitive and Metacognitive Processes in Computer-Based Learning Environments'. In: *International Handbook of Metacognition and Learning Technologies*. Ed. by Roger Azevedo and Vincent Aleven. New York, NY: Springer New York, pp. 143–156.

Gog, Tamara van, Halszka Jarodzka et al. (2009). 'Attention guidance during example study via the model's eye movements'. In: *Computers in Human Behavior* 25.3. Including the Special Issue: Enabling elderly users to create and share self authored multimedia content, pp. 785–791.

Goodfellow, Ian, Yoshua Bengio and Aaron Courville (2016). *Deep Learning*. MIT Press.

Hartridge, H. and L. C. Thomson (1948). 'Methods of investigating eye movements'. In: *British Journal of Ophthalmology* 32.9, pp. 581–591.

Holmqvist, Kenneth et al. (Jan. 2011). 'Eye Tracking: A Comprehensive Guide To Methods And Measures'. In.

Hubel, David H. and Torsten N. Wiesel (1974). 'Uniformity of monkey striate cortex: A parallel relationship between field size, scatter, and magnification factor'. In: *Journal of Comparative Neurology* 158.

Kahneman, Daniel and Jackson Beatty (1966). 'Pupil Diameter and Load on Memory'. In: *Science* 154, pp. 1583–1585.

Klatzky, Roberta and Nicholas Giudice (Jan. 2012). 'Sensory substitution of vision: Importance of perceptual and cognitive processing'. In: pp. 162–191.

Knight, Bruce, Mike Horsley and Matt Eliot (Dec. 2014). 'Eye Tracking and the Learning System: An Overview'. In: pp. 281–285.

Land, M., Neil Mennie and Jennifer Rusted (Feb. 1999). 'The Roles of Vision and Eye Movements in the Control of Activities of Daily Living'. In: *Perception* 28, pp. 1311–28.

Leyba, J. D. and Jimi Malcolm (2004). 'Eye Tracking as an Aiming Device in a Computer Game'. In.

Mangeloja, Esa (2019). 'Economics of Esports'. In.

May, James G. et al. (1990). 'Eye movement indices of mental workload'. In: *Acta Psychologica* 75.1, pp. 75–89.
McCulloch, Warren S. and Walter Pitts (1943). 'A logical calculus of the ideas immanent in nervous activity'. In: *The bulletin of mathematical biophysics* 5.
Mitchell, T.M. (1997). *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill. ISBN: 9780071154673. URL: https://books.google.no/books?id=EoYBngEACAAJ.
Newzoo (2021). *Free 2018 Global Esports Market Report*.
Oyster, Clyde W (1999). 'The human eye'. In: *Sunderland, MA: Sinauer*.
Pedregosa, F. et al. (2011). 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12, pp. 2825–2830.
Rayner, Keith (1998). 'Eye movements in reading and information processing: 20 years of research.' In: *Psychological bulletin* 124 3, pp. 372–422.
Russell, Stuart J. and Peter Norvig (2009). *Artificial Intelligence: a modern approach*. 3rd ed. Pearson.
Salvucci, Dario and Joseph Goldberg (Jan. 2000). 'Identifying fixations and saccades in eye-tracking protocols'. In: pp. 71–78.
Seeber, Kilian (Mar. 2013). 'Cognitive load in simultaneous interpreting: Measures and methods'. In: *Target* 25.
Smith, J. David and T. C. Nicholas Graham (2006). 'Use of Eye Movements for Video Game Control'. In: Hollywood, California, USA: Association for Computing Machinery.
Tobii (2017). *Eye Tracking in Gaming, How Does it Work?* Accessed: 2021-10-17.
Ware, Colin and Harutune H. Mikaelian (1986). 'An Evaluation of an Eye Tracker as a Device for Computer Input2'. In.
Zemblys, Raimondas (Nov. 2016). 'Eye-movement event detection meets machine learning'. In.
Zemblys, Raimondas, Diederick C. Niehorster et al. (Feb. 2017). 'Using machine learning to detect events in eye-tracking data'. In: *Behavior Research Methods* 50.
— (2018). 'Using machine learning to detect events in eye-tracking data'. In.