

# Improving Browser Watermarking with Eye Tracking

NIKOLAI-IRAJ SANAMRAD, University Tübingen, Germany

The problem of authenticating online users can be divided in two general sub problems: confirming two different web users being the same person, and confirming two different web users being not the same person.

In order to improve user authentication techniques in cases, where cookies or logging in is not applicable as well as to actually distinguish users from each other without any authentication, browser, system and hardware fingerprinting is being used more often. This technique improves the ability to distinguish between devices being used, however, the main goal of most online service providers is to distinguish between end-users instead of their devices, even though it is safe to assume that most of the time only one person uses a certain device.

The easiest and most accessible method for fingerprinting used by online services is browser fingerprinting. Browser fingerprinting distinguishes between user devices using information about the browser and the system, which is usually provided by most browsers to any website, even when cookies are turned off. This data is usually unique enough even for identical devices, since it heavily relies on device usage by the user. In this paper, browser fingerprinting is being improved with the usage of information acquired from an eye tracker.

# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
<b>2</b>	<b>Fundamentals</b>	<b>11</b>
2.1	Decision Trees . . . . .	11
2.2	Support Vector Machine . . . . .	13
2.3	The Eye . . . . .	16
2.4	Eye Movements . . . . .	22
2.5	WebGazer . . . . .	25
2.6	Browser Eyetracking . . . . .	28
<b>3</b>	<b>Method</b>	<b>29</b>
3.1	Website for the Study . . . . .	31
3.1.1	Structure . . . . .	32
3.1.2	Functionality . . . . .	34
3.1.3	Process . . . . .	39
3.2	Data Gathering . . . . .	40
3.2.1	Process . . . . .	40
3.2.2	Grid . . . . .	41
3.3	Evaluation with MatLab . . . . .	42
3.3.1	Process . . . . .	42
<b>4</b>	<b>Evaluation</b>	<b>43</b>
<b>5</b>	<b>Discussion</b>	<b>53</b>
<b>6</b>	<b>Conclusion</b>	<b>57</b>



# 1 Introduction

Differentiating users between each other [FBK20] online has always been a relevant issue for a certain category of web applications that specializes in delivering different, personalized content to different users [ZC16], offering the best experience for every user individually, as well as displaying targeted advertisements and, in some cases, sharing information about user activity with third-party services for various reasons, such as improved targeting for advertisements as well as simple activity analysis [ZC16] used to adjust services offered to a certain customer [RFZ01]. Knowing, whether a particular user actually is who the user's device claims, is critical in some cases, such as Online Banking, and has a very high influence on profits in many fields, such as flight ticket sales, where airlines attempt to adjust ticket prices based on almost all possible variables available about the user, as well as the user's recent search history, in order to offer each user the opportunity to book a ticket at a price acceptable for that user, allowing all people who want to book a flight – to book a flight, while managing to bill them differently [ZC16, RFZ01]. Internet retail companies heavily rely on user metadata provided by third-party services too, since they can use it for an improved website layout for each user, a simple example are most of the large online retail websites that look different on different computers, even when no-one is logged in. Apart from online services that actually bill people, a lot of other services, such as news websites, entertainment websites and generally most of the very large websites that are capable of offering different types of content to a user based on their previous activity make extensive use of information about each user regardless of that user being registered, logged in or not logged in, which improves the experience for the user and increases possible monetizing opportunities for those websites [ZC16, RFZ01].

Usually a login-password pair [ZJ12], that has to be typed in by the user, or cookies, which are automatically generated for the user, are used to authenticate a certain person on most websites offering personalized content, but in some cases this approach has flaws, for example, when a user changes devices, uses a different browser or simply turns off cookies, in other cases it is impossible at all or introduces significant overhead for the user, specially in common use cases where a user would not expect to have to log in with personal credentials [ZJ12]. A great example of attempts to find a proper balance between letting the client use some services without logging in while relying on cookies and forcing the client to log in while risking losing the client because of the client's unwillingness to register or log in can be seen on some small business websites selling inexpensive items

and offering simple services, where an item or service can be bought without a registration process, replacing the login-password pair with cookies, which has a main disadvantage of non-transferability between different devices and browsers of the same client [SCM<sup>+</sup>09]. Small business websites usually have readily available competition that offers an alternative for the client, which is easy for the client to switch to, if the client is unhappy with the experience for any reason, which might include the requirement to log in [SCM<sup>+</sup>09]. This forces small business websites to eliminate anything that could potentially make a client unhappy and eventually force him to switch to a competitor. Large internet service and retail companies have the advantages of having virtually no competition and therefore no alternatives for the client, which is why they are more flexible in requiring the client to log in without risking to lose the client, which allows a proper login-password pair to be used for the client and therefore offering more information about the client based on client activities on all client devices [ZJ12, SCM<sup>+</sup>09].

The problem of authenticating online users can be divided in two general sub problems: confirming two different web users being the same person, and confirming two different web users being not the same person. In the past, IP addresses have been considered a reliable method of authenticating online users [GHJP00]. IP addresses are still used quite often to confirm two people being the same person if the two IP addresses match on smaller websites and online services with smaller audiences [GHJP00]. IP addresses cannot be used to confirm two users being different people anymore, because users change their IP addresses more often than devices, for example, while driving and using a phone browser with mobile communication connectivity that switches between cell towers, which quite often changes the IP address [GHJP00]. Since nowadays users change locations, devices and internet service providers multiple times every day, and some web services are used by a large enough percentage of the population to make the likelihood of two different users with the same IP address using the service [TGS<sup>+</sup>14]. Usage of IP addresses can't provide a reliable method of distinguishing users with a high enough precision. In order to improve user authentication techniques in cases, where cookies or logging in is not applicable as well as to actually distinguish users from each other without any authentication, browser, system and hardware fingerprinting is being used more often [TGS<sup>+</sup>14]. This technique improves the ability to distinguish between devices being used, however, the main goal of most online service providers is to distinguish between end-users instead of their devices, even though it is safe to assume that most of the time only one person uses a certain device. The easiest and most accessible method for fingerprinting used by online services is browser fingerprinting [BFGI11]. Browser fingerprinting distinguishes between user devices using information about the browser and the system, which is usually provided by most browsers to any website, even when cookies are turned off [BFGI11]. This data is usually unique enough even for identical devices, since it heavily relies on device usage by the user. In this paper, browser fingerprinting is being improved with the usage of information acquired from an eye tracker [KSF<sup>+</sup>15, Fuh19, FBK20, FTBK16, FKS<sup>+</sup>15b]. Usually

the gaze estimation is acquired using the pupil center [FSR<sup>+</sup>16, FKH<sup>+</sup>17, FGS<sup>+</sup>18, FGK20b, FSK17b, FEH<sup>+</sup>18] and computing a mapping function between the pupil center and the screen [FGK20a]. For this multiple techniques have been proposed where currently the semantic segmentation [FGRK19, FRK19, FCZ<sup>+</sup>18] is used. While the eye tracking signal contains a lot of information about the cognitive state [ESF<sup>+</sup>17, EFK17, EHF<sup>+</sup>17, BFG<sup>+</sup>16] or the tiredness of a person [FSG<sup>+</sup>16, FSG<sup>+</sup>17, FSK17a], we only used the gaze information alone in this pilot work. We used Webgaze [PSL<sup>+</sup>16a], which works entirely in the browser and enables to estimate the gaze of a person and therefore the salient regions [GFSK17, FKSK18, FKB<sup>+</sup>18, FKS<sup>+</sup>15a]. Since many studies have shown that a scanpath [FBH<sup>+</sup>19, FCK<sup>+</sup>19] is a unique biometric feature for every person, each user's scanpath which consists of eye movement types [FRE20, FSK<sup>+</sup>18, FCK18a, FCK18b, FK18, FRE20] is being saved and compared to other scanpaths. The main problem of the usage of a scanpath is that it creates a large amount of overhead for the user, including initial calibration and the requirement to keep the head visible to the web camera, and actual access to the web camera, therefore it is unlikely to be used with most of the current web services, however, the detection of a driver's gaze is being implemented in the automotive industry, and if the gaze-detection technology will improve further, it might start being used in mobile devices and computers [Fuh20], similar to fingerprint sensors being available in many modern smartphones. Such an implementation could offer the gaze point to a wide array applications, with the initial goal of improving user experience, but it could also be used for further user authentication.



## 2 Fundamentals

In order to have a clear understanding of what is being done in this study, several concepts and definitions, as well as overall techniques and systems are required to be introduced first. In this chapter, the necessary definitions, concepts, machine learning techniques [FKRK20, ?, FK20b, FK20a] and systems are described.

### 2.1 Decision Trees

In general, a decision tree is a tree-like structure used to make a decision about a certain object based on the object parameters 2.1. Decision trees are used in many fields. in this experiment, the relevant type of trees are the decision trees used in machine learning.

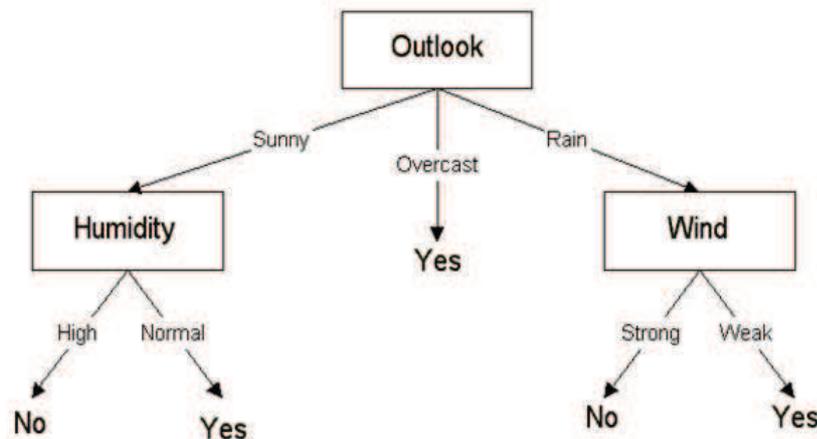


Figure 2.1: An example of a simple decision tree, tasked with making a decision based on weather [Kul17]

Decision trees in machine learning are usually mathematical models used to make an assumption about a certain object, which is usually represented by a vector, based on assumptions about previously processed similar objects. The process of processing similar objects is called "learning" or "training" of the tree model, and the similar

## Chapter 2. Fundamentals

objects used for training are called the "training dataset". For example, the training dataset for the decision tree on Figure 2.1 is displayed on Figure 2.2.

Outlook	Temperature	Humidity	Wind	Play Golf
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Figure 2.2: The training dataset used for the decision tree displayed on Figure 2.1 [Kul17]

In order for a machine learning decision tree mathematical model to be grown, the training dataset is being analysed in order to make the most optimal decision tree using the relevant parameters of an object and eventually ignore non-relevant parameters. There are multiple algorithms capable of building a decision tree. Most of the algorithms use the "entropy" and "information gain" characteristics, which are calculated for each parameter in the dataset, based on the actual values in the dataset, in order to decide, where to use each parameter in a tree, and whether to use a parameter in the tree at all, in order to avoid overfitting the trees [Wik20j, Wik20i]. Entropy is the characteristic of a parameter in a decision tree evaluating the randomness of the information in that parameter for the decision. The higher the entropy of a certain parameter, the less likely it is going to be used close to the root of the tree. In some decision tree growing algorithms, entropy is calculated for each possible set of parameters below the parameter that is being analysed for entropy. The entropy is used in the calculation of information gain, which is used to calculate, how much each parameter of the training set would contribute to the final decision or classification. Usually the Algorithm growing a decision tree chooses the parameter of the dataset with the highest information gain value to be used closer to the root, or even as the root. Apart from the described characteristics, various decision tree growing algorithms use various additional parameters aimed at reducing the tree size and optimizing the tree. For example, some characteristics are used to "prune" the tree to avoid overfitting. Pruning describes the process of



would effectively serve as a threshold for distinguishing between these two classes. As an example, a one-dimensional space with two classes of samples distributed across the line in two groups would have the threshold between those two groups, as shown on Figure 2.4. The threshold would be a dot in a one-dimensional space, and a dot is a hyperplane for a one-dimensional space. The threshold is usually located in the middle of the closest elements of each class [Gun20].



Figure 2.4: A margin classifier [Gun20]

The threshold could be used to classify future samples based on their location relative to that threshold. In order to compensate for the outliers affecting the classification as shown on Figure 2.5, a margin classifier could have a safety or soft margin.



Figure 2.5: A margin classifier being affected by an outlier (the red circle close to the green circles) causes the new black sample to be classified as red despite being closer to the green group [Gun20]

In order to avoid outliers affecting the threshold, a soft margin is calculated by cross-validating all training samples to find the optimal location for the new margin called the support vector, as seen on Figure 2.6, which is calculated to have as little false detections as possible.



Figure 2.6: The support vector [Gun20]

After the support vector has been calculated, new samples can be classified correctly based on their location relative to most of the samples in each class, as shown on Figure 2.7.

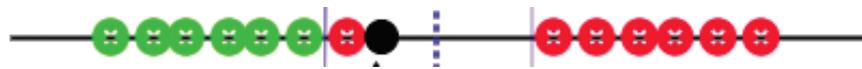


Figure 2.7: The new black sample will be classified as green because of the support vector being calculated to be between the majority of the elements of each class [Gun20]

The support vector solves the problem of outliers, but does not solve the problem of two groups being distributed among a space without a clear support vector position, as shown on Figure 2.8.



Figure 2.8: Two groups of samples are distributed in a fashion not allowing for a hyperplane to divide them in a one-dimensional space [Per]

In order to make the division by a hyperplane of such examples possible, Support Vector Machines were introduced [CV95]. Support vector machines, in case of two groups of samples being distributed in a way that would not allow their division by a support vector in their own dimension, search for a hyperplane in a higher dimension, by using a kernel function to elevate the samples into a higher dimension, as shown on Figure 2.9. In a two-dimensional space, a hyperplane is a line, which is calculated by cross-validating each of the samples with the rest in order to find the optimal position for the hyperplane dividing the two groups [Per, Gun20].

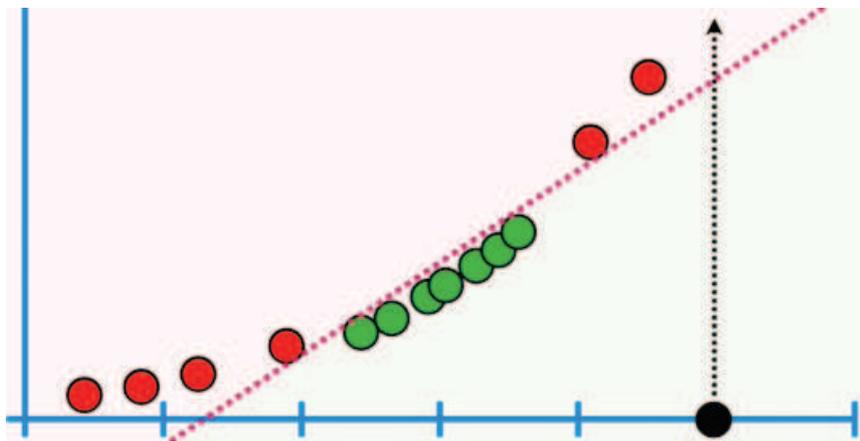


Figure 2.9: The samples from Figure 2.8 have been elevated into a higher dimension, with a new sample being correctly classified, and a hyperplane dividing the two groups [Per]

When the samples are lifted to their higher dimension by having their next dimension calculated based on their current dimension with a kernel function, a hyperplane is being calculated just like a support vector would. In order to reduce computational complexity, usually the samples are not actually lifted to their higher dimension, instead the calculations to find the hyperplane dividing the classes are made based on the kernel function of their current dimension, and for each new sample that needs to be classified, the higher-dimensional position of the new sample is calculated using the kernel function, which is sufficient to find the location of the new sample relative to the hyperplane dividing the groups [Per, Gun20, Wik20z].

## 2.3 The Eye

This study is highly concentrated on eye movements, and therefore the eye itself is important. A human eye could be seen as a module in a human body responsible for sensing the visual spectre of the electromagnetic field, in other words, it is used to see light and distinguish different colours of light, and different colours of light are just different frequencies of the visible spectrum of the electromagnetic field (Figure 2.10).

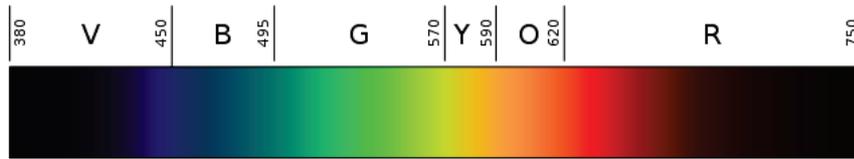


Figure 2.10: The visible spectrum of light, with the wavelength in nm [Wik20q]

Human eyes are generally capable of sensing electromagnetic waves with wavelengths between 380nm and 740nm[Sta06], so the visible range of the spectrum for humans is only 360nm wide. A 360nm wide area of the electromagnetic spectrum is a relatively narrow part of the whole spectrum (Figure 2.11).

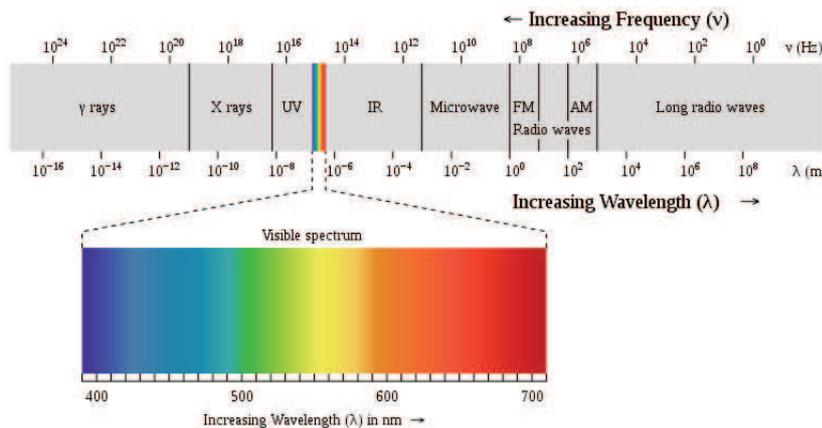


Figure 2.11: The whole spectrum of electromagnetic waves [Wik20q]

Although the visible part is narrow compared to the whole electromagnetic spectrum, the ability of human eyes to sense and provide humans with information about electromagnetic radiation with different wavelengths is vital for not just humans, but most of the animals and is being constantly used, sometimes providing humans with critical information (Figure 2.12).

Humans rely on their eyes on a daily basis, from the moment they wake up to the very last minute they are still awake. Pretty much any interaction a human



Figure 2.12: Traffic lights provide critical information to humans operating motor vehicles via the visible part of the electromagnetic spectrum [Wik20b]

participates in involves the functionality of eyes, that's why most humans have two eyes at the front of their head, since it's a perfect location for eyes to collect information about their environment and have the best possible outlook because of a high location on the body. The location of human eyes also provides them with more robustness, since the connecting nerves between eyes and the brain are short and therefore are less susceptible to mechanical failure. Additionally, the shorter nerves connecting the eyes to the brain make the delay between the signal being sent and received unnoticeable.

The part of the eye, that actually senses light, is called the retina and is a layer of sensors covering 72% of the the rear side of the eye (Figure 2.13).

The sensors are called rods and cones and generally are photosensitive sensors converting light into signals, which they detect using a chemical reaction that happens inside the photoreceptor proteins. The chemical reaction causes the cell membrane to change the membrane potential, which is sensed by the nerves [Wik20u, Wik20v].

Although light is being sensed by the rods and cones, they have a layer of axons in front of them, which light first has to pass through. The first layer, that light passes through, is the ganglion layer containing axons from the optic nerve, after which it reaches the rods and cones. Because the layer connected to the optic nerve is located above the layer of rods and cones (Figure 2.14), and the optic nerve is located at the opposite side of the layer with rods and cones, the eye contains a "blind spot" – an area where the ganglion layer with axons connects to the optic nerve [Wik20u, Wik20v].

As shown in Figure 2.13, the eye has more components than just the retina. The outermost layer of the eye is the structural layer supporting the shape of the eye is called sclera (Figure 2.13). The eye is filled with two different fluids – aqueous humour and vitreous body. The vitreous body fills 80% of the eyeball between the retina and the lens, has various viscosity in different areas of the eye and changes over the lifetime of a human [?]. The space beyond the lens is filled with a different

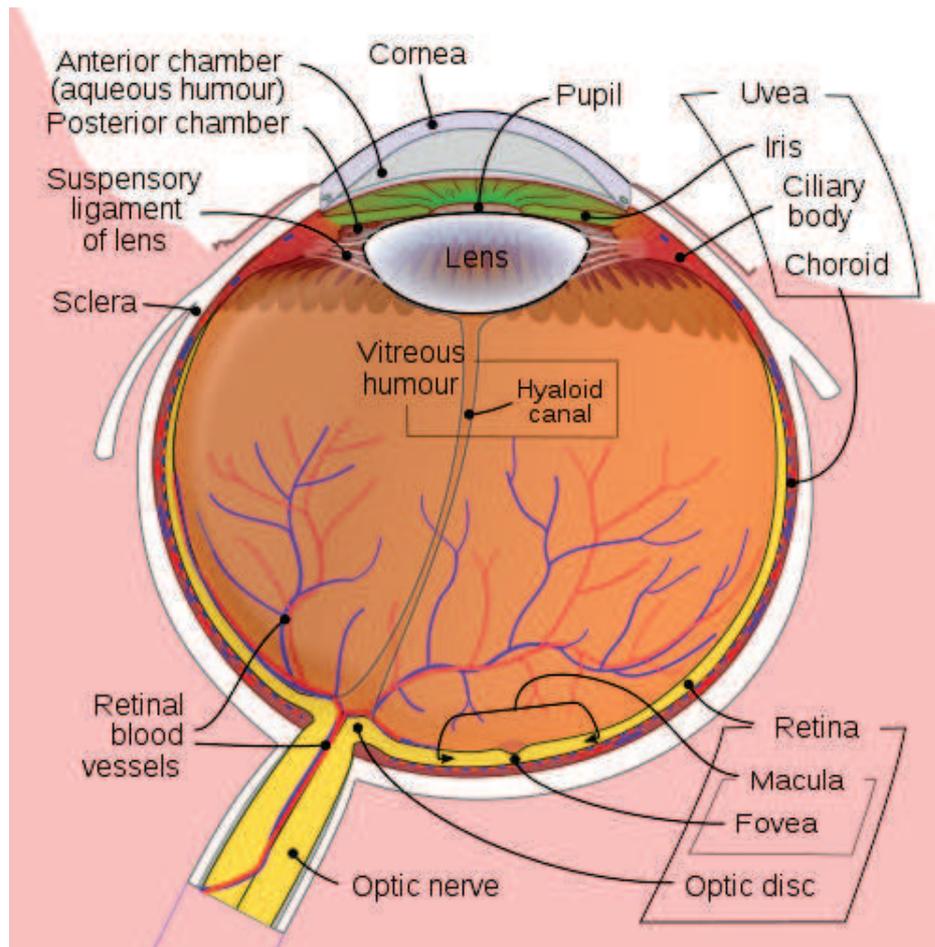


Figure 2.13: The general structure of a human eye [Wik20o]

fluid – aqueous humour. Aqueous humour consists of 98% water [Wik20c]. The lens, the iris and the pupil adjust brightness and the image properties (Figure 2.13) [Wik20o].

The light, properly adjusted, is absorbed by the photoreceptor cells – the cones and rods in the retina. There are two different cells – rods and cones, rods are much more sensitive and are capable of sensing even one photon [OS07], however, rod cells offer very little colour sensitivity. There are 120 million rod cells in the eye [Sch11] and most of them are located close to the outer edges of the retina, which allows them to perform great for peripheral and night vision [Wik20u, Wik20w]. The cone cells are more interesting, since they sense some colours more intensively than others. Cones are, compared to rods, much faster, but have a lower sensitivity [Wik20g]. There are three types of cone cells: S, M and L. Their names come from the light they are sensitive to: short-wavelength light, medium-wavelength light and long-wavelength light. Cones on their own cannot detect colour, they just sense different wavelengths

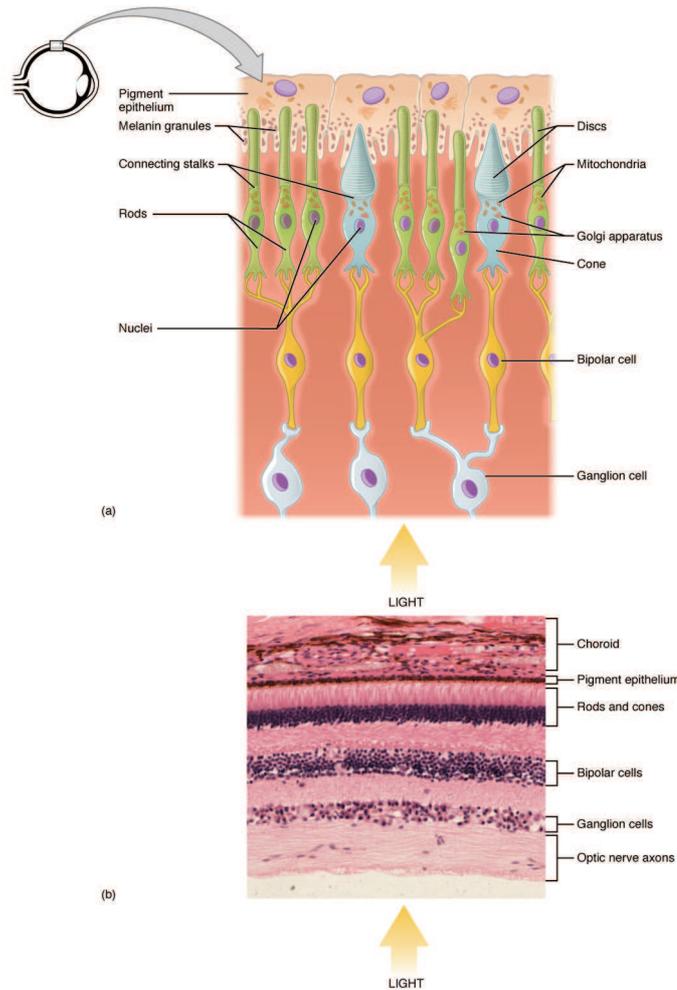


Figure 2.14: Different layers of the 0.5mm thick retina [Wik20v]

with a different intensity, with a certain wavelength causing the highest sensing intensity for cones. The S-cones have their peak sensitivity at 420nm-440nm, the M-cones have their peak sensitivity at 534nm-545nm, the L-cones have their peak sensitivity at 564nm-580nm [Wik20f]. As shown on Figure 2.15, each type of cones senses a certain wavelength with the highest intensity, while the intensity of sensing colours of other wavelengths in close proximity to the peak wavelength is lower. The cell only receives a signal intensity, so for example, if the top signal intensity of an M-cell is at 540nm, at which it receives the highest intensity signal, for example, at 95%, if a different colour is sensed and passed through to the optical nerve, with sensing a signal intensity of an unknown colour at around 90%, the cone cell is not capable of sensing the exact wavelength. It only senses a signal representing, as an example, either 520nm or 550nm. In order to determine the exact colour or wavelength, the signal from a different cone-cell is required, for example, an L-cell,

which could have, for example, a much lower intensity at 80%, while the M-cell would have an intensity of 90%. Based on the difference, the brain determines that the wavelength is 520nm. This process doesn't take place in the photoreceptor cells, in the retina, in the eye or in the optical nerve. This process takes place in the brain, where only the intensity levels are taken by the optical nerve [Wik20u, Wik20f].

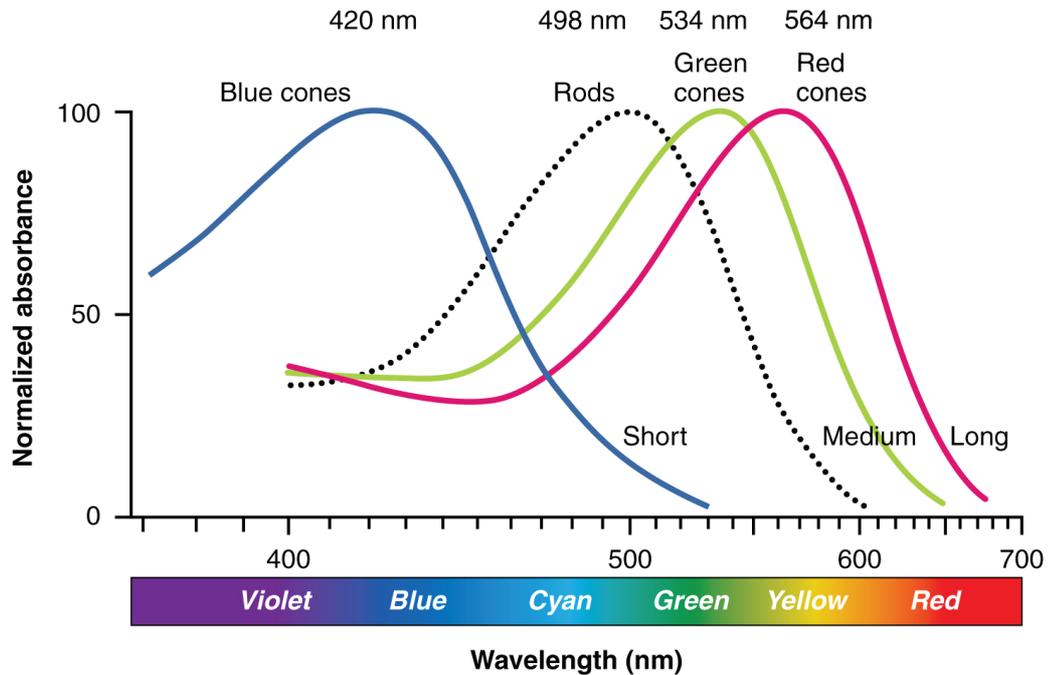


Figure 2.15: Sensitivity intensity of different cones to different wavelengths [Wik20u].

The eye, being a consistent organ shaped like a sphere, with an optic nerve and blood vessels connected in the proximity of the blind spot, is flexible and capable of rotating to a certain extent inside the orbit in order to make the pupil face a desired scene even if it's not directly in front of the human [Wik20o, Wik20t].

The eyes are located inside their orbits – the cavities in the bone where eyes are expected to be, with a layer of fat (Figure 2.16) [Wik20o].

There are six muscles actuating eye rotation around each of the three axes in opposite directions. The medial rectus rotates the eye around the vertical axis inwards and the lateral rectus rotates the eye around the vertical axis outwards [Wik20r, Wik20d, Wik20l]. The superior rectus rotates the eye upwards and the inferior rectus rotates the eye downwards [Wik20d, Wik20l]. The superior oblique rotates the eye around the horizontal axis inwards, and the inferior oblique rotates the eye around the horizontal axis outwards [Wik20d, Wik20l]. These muscles allow for the eye to be effectively rotated towards any object in front of a human (Figure 2.17).

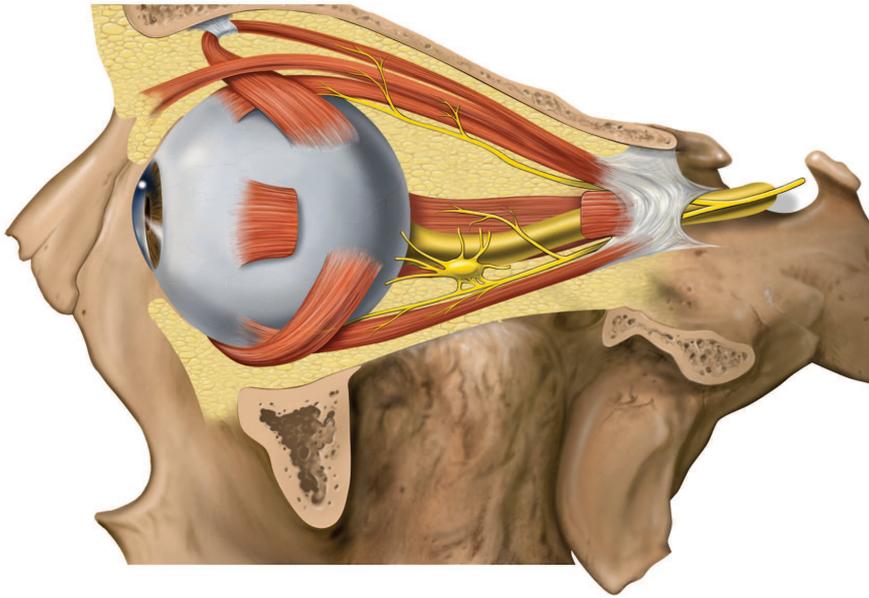


Figure 2.16: Location of the eye inside it's orbits, with muscles and a layer of fat as well as the optical nerve being visible [Wik20o]

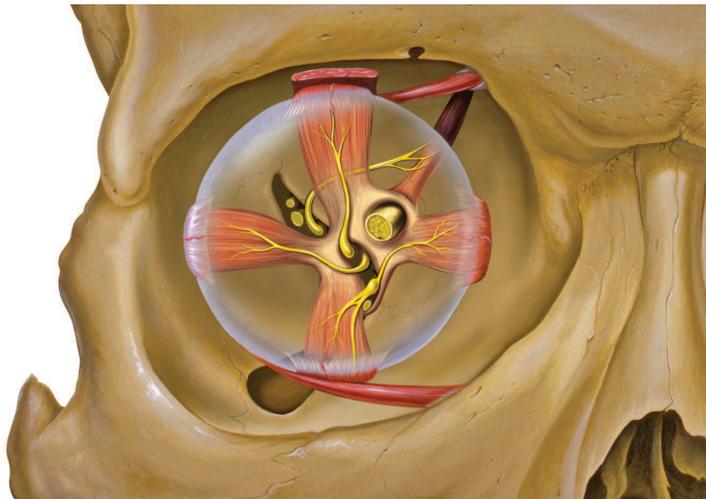


Figure 2.17: Eye muscles allow the eye to be rotated in any direction [Wik20o]

In order to achieve high signal accuracy and acuity in the focused area of the eye, a special area of the eye is filled exclusively with cone cells. This area - the fovea, offers the highest accuracy (Figure 2.13) despite taking only 1% of the retina. In the fovea, each cone feeds to at least one axon connected to the optical nerve and the Fovea itself accounts for 50% of the optical nerve. Since the Fovea doesn't have any

rods, it's not very suitable for operation in low light conditions. It's usually used for reading, driving and other tasks requiring high image quality [Wik20n].

Overall, the majority of humans have two eyes. The two eyes are located at a high point on a human, on the face, above the nose but under the forehead, in their special sockets in the bone – orbits. Eyes have a close to spherical form, with 6 muscles attached to them, allowing the eyes to be rotated around all of the three axes, with each muscle being responsible for rotation in one direction around each of the axes. Eyes have all their interfaces with the rest of the body – the optical nerve carrying the signals from the photoreceptor cells, as well as the blood vessels, connect and enter the eye at one point – the blind spot, which permits higher flexibility for eye rotation. The eye itself consists of multiple optical components adjusting the light brightness and focus, acting as a lens. The light signal is being detected by photoreceptor cells – rods, offering great low-light performance and cones, offering high precision, located on the Retina, which covers the rear side of the eye. Since the nerves are wired above the photoreceptive cells, the connection point of the optical nerve to the eye has the layer of cones and rods pierced by the optical nerve, which is why the connection point is called the blind spot. Unlike the blind spot, the fovea is an area on the retina offering the highest accuracy and image quality, which is why the focused image is usually projected on the fovea. The signal from the retina gets transferred to the brain through the optical nerve. Eyes are vital to humans and are being used almost constantly during human activity.

### 2.4 Eye Movements

In order to have a clear understanding of the subject of this study, it is important to have an exhaustive understanding of eye movement origins, mechanics and communications. Eye movements, similar to most fast movements in the body of a human, are actuated by muscles (Figure 2.18). The six muscles consist of three pairs of antagonist muscles rotating the eye around all available axes to a human. As an example, when a human attempts to rotate eyes upwards or downwards, the superior rectus and the inferior rectus are involved (Figure 2.18), so that in order to rotate the eye upwards, the superior rectus is actuated, and to rotate the eye downwards, the inferior rectus is actuated [Wik20m]. All eye muscles rotate the eye in a similar fashion – the antagonist pairs rotate the eyes in opposite directions around an axis.

The eye muscles are controlled mainly by the oculomotor nerve, but the superior rectus is controlled by the abducens nerve and the lateral rectus is controlled by the trochlear nerve, all of which are cranial nerves (Figure 2.19) [Wik20m, Wik20s, Wik20a].

These cranial nerves come straight from the brain, which allows for fewer points of failure, low latency and direct connectivity between the brain and certain organs in

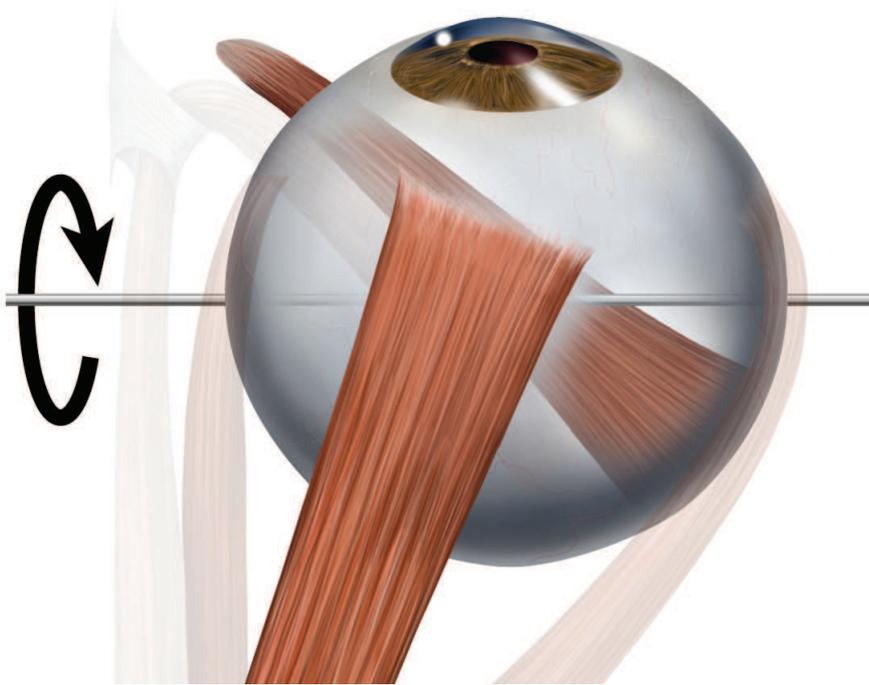


Figure 2.18: Eye from above, the superior rectus in the foreground and inferior rectus in the background, the pair of antagonist muscles responsible for the ability of an eye to look upwards and downwards [Wik20y]

close proximity to the brain, including the eyes [Wik20h].

The way a human brain controls eye movements by actuating eye muscles is shown in Figure 2.20. Since the eye muscles are antagonist muscles, each muscle has to be controlled solely to hold position or to contract. In order to contract, the neurons fire into the necessary muscle with a certain rate. In order to hold position, the neurons fire with a steady, baseline rate. The longer the initial burst of firings happens, the further a muscle contracts and therefore, the further an eye rotates. The further the eye is rotated, the higher the required baseline frequency of firings for the eye to hold position is. Without the baseline firings while an eye is rotated, the eye will rotate back to its resting position [PD01].

In general, the human eye interacts with objects by scanning them. The eye movements include fixations and saccades, where fixations are periods where the eye is fixated on a certain point, and saccades are periods where the eye gaze changes rapidly from one position to another. Usually saccades are short and fixations are long. The patch left behind by saccades and fixations is called the scanpath. The

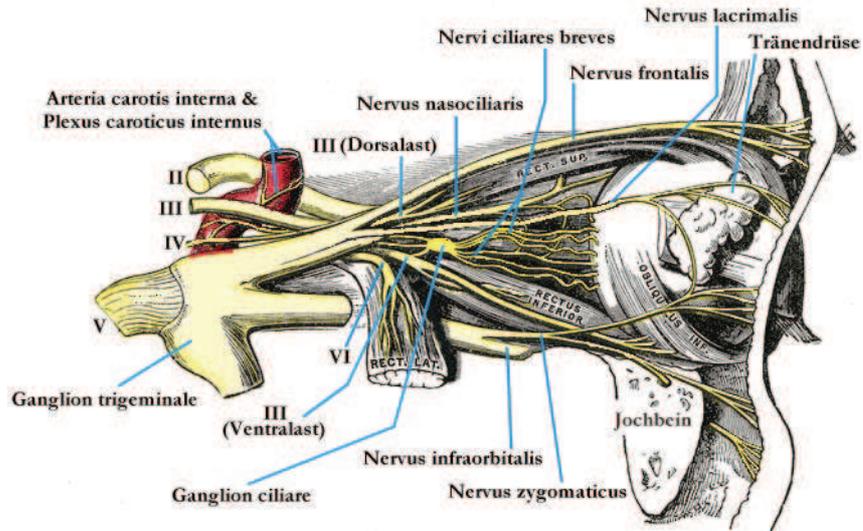


Figure 2.19: Nerves around the eye [Wik20x]

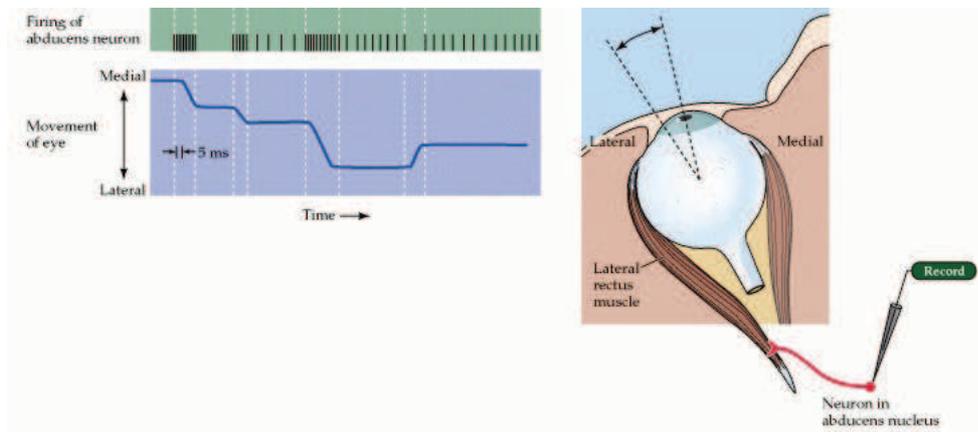


Figure 2.20: The neuron bursting rate is depicted as the frequency of black lines corresponding to eye movement. The baseline frequency can be seen with the eye holding position (horizontal line in the blue area), while the movement distance is clearly correlated to the length of a bursting event. [PD01]

scanpath is a unique feature of each individual, similar to a fingerprint, however, unlike a fingerprint, a scanpath involves a temporal attribute (the time of each fixation, or, more precise, a sequence of fixations) which makes it more robust towards being compromised compared to a fingerprint, since it requires an observation being made over time, and can't be documented at a later stage, which a fingerprint is susceptible to. Additionally, even though a large percentage of devices is being equipped with a fingerprint sensor, the majority of devices is equipped with a

front-facing camera, making more devices capable of capturing a user scanpath, which additionally is less invasive into user privacy compared to the fingerprint data, or even a picture of the face of the user, since it essentially requires just the sequence of user fixations to be analysed and is being extracted on the user device, preventing any leakage of actual pictures of the user, if implemented properly. In this experiment, we concentrate most of our attention at saccades – even though fixations and saccades are non-exclusive, and fixations with saccades can be derived from each other, and we extract saccades from fixations in this experiment, the main datapoints are saccades – the movements of eyes from point A to point B, which we use for our analysis.

The movements of the eye relies on two variable factors across humans – psychological and physical. The psychological aspect involves the way the brain controls the eye, whether both eyes are controlled similarly and are given the same control signal, or whether they are being given different signals depending on their position. The physical aspect of eye movements involves the way each eye, as a separate, physical light-sensing module, responds to the control signals and how it moves. The physical aspect also involves the development of certain eye muscles, since the development of eye muscle groups directly influences the way an eye as a physical module would respond to a signal from brain asking it to face a certain direction, or to stop facing a certain direction. The psychological aspect of eye movements involves the actual signals a brain sends to the eye module, and the parameters of those signals. Previous studies have shown that personality traits/differences have an influence on the way humans scan scenes they see [HH16, KV11, HLMB18a, PJD10, HLMB18b] and in general, eye movement patterns are different for different individuals [BPS72, STD19, NL15] and generally might depend on the difference of developmental processes between individuals [CCH<sup>+</sup>17] and actually change as individual humans develop over time [OLSM07, LVG08]. All this implies that eye movement patterns are highly individual and could be used as a psychometric, since they develop over time, as an individual develops. In fact, many studies have shown the possibility of individual authentication based on the eye movement pattern – the scanpath, as a biometric feature [CGN<sup>+</sup>15a, KO04, Kas04a, FMSM09, MFS04, WMH11, RPTH17] and a lot of different patents have been issued for eye movement authentication [BH07, JCG17, Her15, MS16]. All these applications of eye tracking movement recordings make use of the property of individual eye movement patterns depending on individual characteristics such as eye muscle development, psychological features as well as just differences in interest for different areas on a stimuli.

## 2.5 WebGazer

One of the most important tools for collecting user gaze data used in this study is WebGazer [PSL<sup>+</sup>16b], a browser-based eye tracker offering impressive accuracy for a browser-based eye tracking tool written in JavaScript, which simplifies the study

making it possible for anyone with a modern browser and a webcam to participate in the study. WebGazer is capable of estimating the gaze point using two different methods. The first method estimates the gaze by detecting the pupil and mapping the pupil position to the known gaze locations on the screen. The second method applies a ridge regression model to the input data based on continuous calibration input.

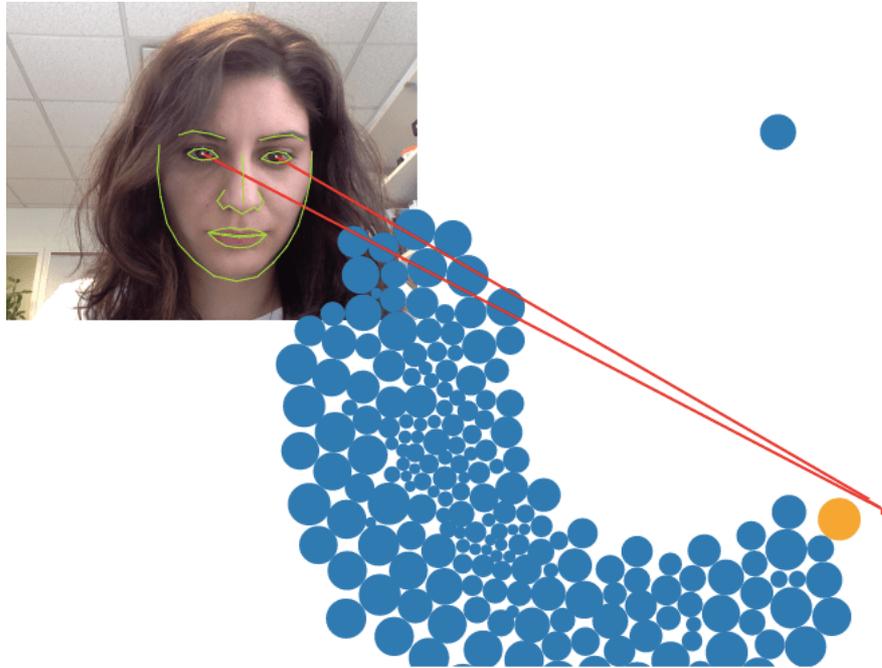


Figure 2.21: An example of WebGazer tracking the gaze on a website. [AP]

Since it has been shown, that cursor movements and mouse clicks usually indicate the location of the gaze of a user with acceptable tolerances [HPW11a, CAS01, GA10, LD14, RFAS08], WebGazer utilizes user clicks and mouse movements in order to gather calibration information, which is readily available in modern browsers. Calibrating WebGazer with user interaction data such as mouse clicks and cursor movements allows the separated calibration part, which is required for most eye trackers, to be avoided entirely, which makes the usage of WebGazer on websites much easier.

Apart from being capable of operating without a distinctive separate calibration stage, WebGazer has all the components and steps of a conventional web-cam based eye tracker. WebGazer can use three different JavaScript libraries [EMD<sup>+</sup>18, EL14, Tsc12] to extract the relevant parts of the webcam picture containing eyes in real time.

The parts of the picture containing eyes are the main part being analysed in order to extract the gaze. WebGazer utilizes two different methods to estimate the gaze in real time based on the eye images from the webcam in real time. The first method

utilizes the pupil location on the picture of the eyes as an input into a simple linear regression model. The second method uses the whole eye image divided into a 6 x 10 grid, with some filters applied and in gray scale, resulting in a 120D vector, which is then used as an input into a ridge regression model [HK12], which maps the 120D vector representing the eye features to an estimated gaze point.

Instead of having a separate calibration stage, like most webcam based eye trackers have, WebGazer self-calibrates. Self-calibration is implemented based on various studies showing the correlation between mouse clicks and cursor movements [HWB12, HPW11a] which allows WebGazer to gather calibration data from user interaction, which is represented by cursor movements and mouse clicks. This allows the calibration to be happening all the time a user interacts with web gazer. Studies showing the correlation between mouse clicks and gaze points on the screen [HWB12] indicate, that the average distance between a user gaze position on a screen and the mouse pointer is 74 pixels at the time of a mouse click event, which is close enough to the actual mouse click position to assume that they are located in the exact same place. Additionally, 500ms of eye samples are kept in a fixation buffer and when a mouse click occurs, the buffer samples are analysed. The samples from the fixation buffer pointing to positions within 72 pixels around the mouse click position are then added to the calibration set. The average duration of a fixation is 200-500ms, and it is safe to assume that an eye fixation on a certain position on the screen has to occur before the mouse is being moved to that certain position, which is why WebGazer analyses the previous 500ms of eye samples for possible calibration samples that could improve accuracy to be added to the model. Additionally to mouse click events, cursor movements are used to gather information about the the user gaze position used for self-calibration in WebGazer. Based on studies showing, that when the cursor is being moved to click, the gaze position correlates to the cursor position [HPW11a], WebGazer uses cursor movement information for self-calibration. Every time the cursor is being moved, the cursor position combined with the corresponding eye samples are being added to the Ridge Regression model, although the weights used for the cursor movement samples added are just half the weights for click events, and when the cursor stops moving, the weights for the samples are still added to the model for 500ms while having their weights for the model linearly decreased.

In general, WebGazer [PSL<sup>+</sup>16b] is an open-source eye tracker capable of avoiding calibration while tracking user eyes in real-time in the browser, which is a perfect tool to be used on websites to improve user experience or authenticate a user based on the scanpath.

## 2.6 Browser Eyetracking

Tracking the scanpath of a browser user on a website has been an interesting topic for many researchers [PSL<sup>+</sup>16b, DB12, MKMS17, RPC01, HPW11b], however a separate specialized device is used frequently in order to track browser-usage gaze patterns produced by users. The usage of a specialized device for eye tracking makes the process of eye tracking more complex, but most of the specialized devices offer a fairly high tracking precision.

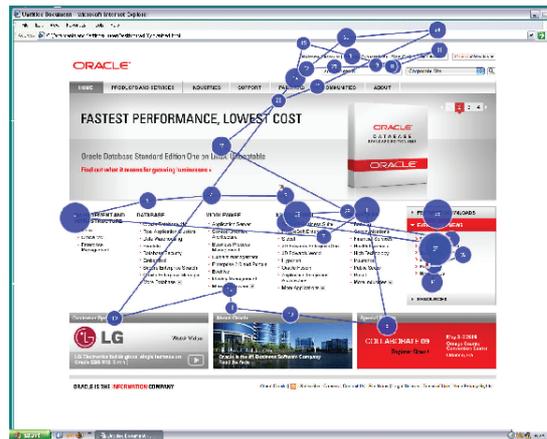


Figure 2.22: An example of a user scanpath in a web browser [GH10]

A large percentage of eye tracking studies involving the usage of a browser by a user are aimed at determining certain regions of interest on one distinct webpage [LW01], or on webpages in general, in order to analyse the behavioural part of the interaction with a webpage, as well as whether or not some methods of delivering information to a user is more or less effective.

Most of the setups for web-browser eye tracking consist of a web browser and an eye-tracking system, where the eye-tracking system might be implemented either as a standalone device or as a separate software using the web camera of the computer being used.

### 3 Method

The methodology used in this study is based on a standard browser on a computer with a web camera. In general, any computer with a web camera and a modern browser should be an acceptable data-gathering instrument for this study, which makes it easily scalable. In order to gather general information about individual eye movement patterns, a website has been constructed that would allow the exploration of 5 different websites during the eye tracking process. All 5 websites are differently layed out and have different content, which allows for a more generalized data gathering compared to GANT [CGN<sup>+</sup>15b], where a standardized stimuli with distinctive regions of interest available has been used.

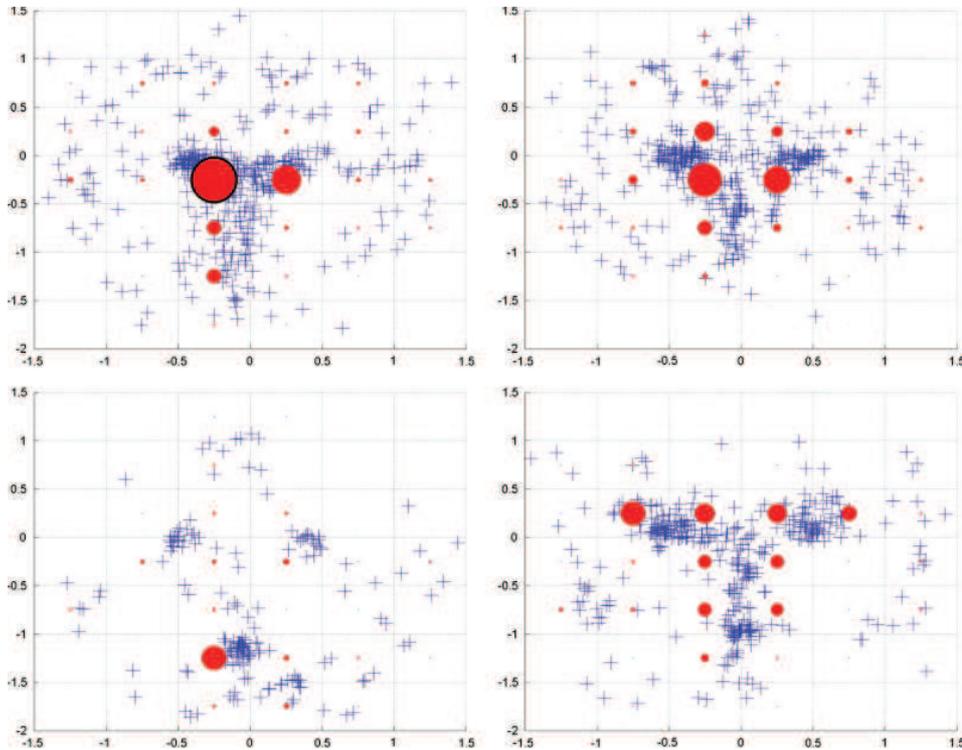


Figure 3.1: An example of fixations on a face stimuli with grid [CGN<sup>+</sup>15b].

Even compared to the study by [MFS04], where an image that has already been seen by a user is used as stimuli, this study utilizes a comparably more generalized approach, since most of the interchangeable pages this study uses as stimuli are

## Chapter 3. Method

actual live online webpages that sometimes change dynamically and have different content during the experiment.

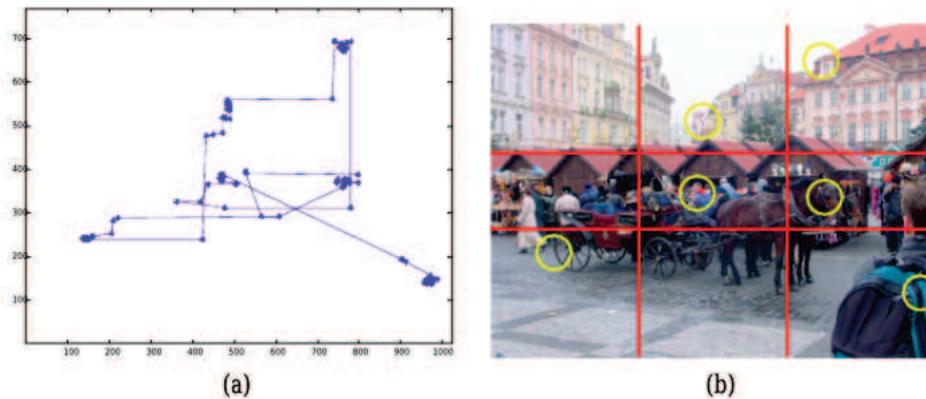


Figure 3.2: Constant stimuli (b) used by [MFS04], where the similarity of a scanpath (a) is used as input data.

Our approach might be in some way similar to the approach used by [Kas04b], where the stimuli was random (Figure 3.2) and the individual eye movement patterns were analysed, since our study offers somewhat random stimuli too, but it uses less precise eye tracking accuracy and a realistic stimuli, which makes it more relevant for actual applications, where perfect conditions for eye tracking usually cannot be provided. Our study can be seen as having an approach close to many previous studies [CGN<sup>+</sup>15b, MFS04, Kas04b], in some cases, even a mix of those approaches in terms of data gathering.

The main difference of our approach is the availability of 5 different websites [17, 18, 19, 20, 21] that a user is allowed to freely switch between at any time during the experiment. Allowing for 5 different websites to be interchangeably switched at any time on user's signal provides us with a general image of an individual user web browsing eye movement pattern and allows for the experiment to be less painful for the user being experimented on. In this study, the 5 websites used have been selected to be as different as possible, while still being interesting for a user. Any website couldn't be picked for this experiment because of technical limitations of some websites not allowing to be shown inside an iframe. Since the self-calibration feature of WebGazer [PSL<sup>+</sup>16b] couldn't be used on web pages inside an iframe because of some rational safety limitations of the iframe functionality, an additional calibration stage had to be added to the experiment, which makes one of the main features of WebGazer – self calibration, almost unused, which is a trade-off for being able to show live websites instead of static stimuli. The self-calibration feature was still capable of analysing user interaction during the interaction of the user with the website interface responsible for interchanging stimuli websites. There are many methods of avoiding the iframe safety limitations allowing websites to be shown

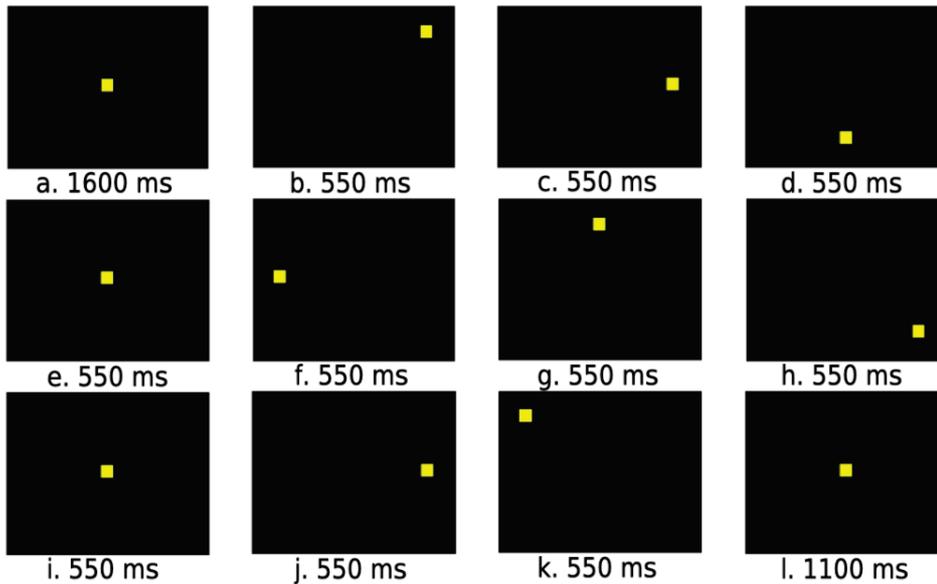


Figure 3.3: An example of random stimuli used by [Kas04b], the number under each stimuli is the time needed for the user experimented on to move the gaze to the position on a screen.

with functioning self-calibration feature, which offers many possible further work opportunities in improving this experiment for further studies.

### 3.1 Website for the Study

In order to perform this study, a simple website using JavaScript, PHP and HTML has been constructed containing most of the main components. The general purpose of the website was to provide a user with all necessary instruments for the experiment, as well as execute the experiment on the user. In addition to the main experimenting tools, a debugging tool page has been made in order to make the observation of the fixations of any id possible, which has been used mainly for debugging and for extended visual information gathering about a certain experiment. The website could be run on any LAMP-compatible web server, local or remote. Because WebGazer [PSL<sup>+</sup>16b] uses getUserMedia API to get the webcam feed from the user, and getUserMedia requires a secure https protocol to be supported on the web server to allow the webcam feed to be passed through, a web server used to run the website has to have supported https protocol support or the safety feature has to be changed in the browser, which is different for all browsers. Since the website has been running on Awardspace [22], only http was available for use, which prompted the study participants to have to change their browser settings in order to allow getUserMedia API to work with http.

The five websites used as stimuli for the scanpath recording experiment were used inside an iframe [24], which has an absolutely rational safety feature – JavaScript cannot record mouse clicks and movements inside an iframe. An iframe is a HTML tag allowing another website to be displayed inside a website, as long as the remote web server of the other website being displayed inside the iframe permits iframe usage by setting the X-Frame-Options [23] parameter in the HTTP header to "allow". Most of the websites selected for the experiment had their X-Frame-Options HTTP header parameter set to "deny" or "sameorigin", which made them unsuitable for this experiment. Despite the X-Frame-Options settings for most interesting websites not allowing them to be used inside an iframe, it was possible to collect a set of interesting websites with different content allowing their usage inside an iframe HTML tag.

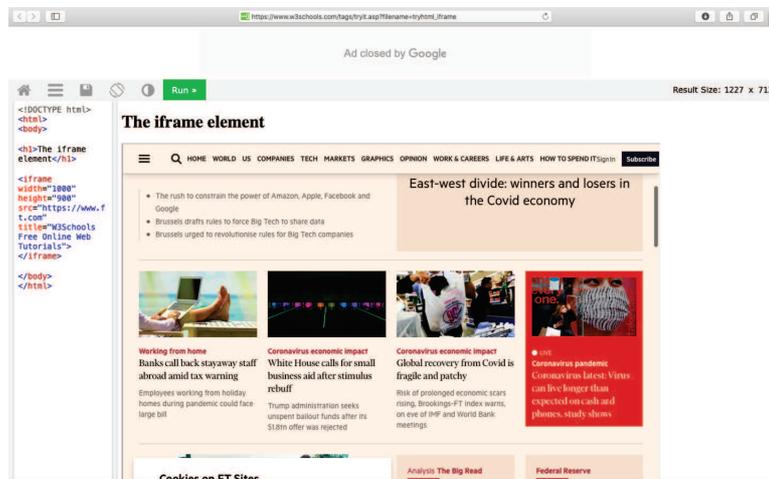


Figure 3.4: An example of an iframe being used. FT.com is being displayed inside an iframe element on www.w3schools.com

### 3.1.1 Structure

The website consists of the main html page which guides the user through the whole experiment. The main page has the calibration pattern, includes WebGazer [PSL<sup>+</sup>16b], the necessary database saving functionality as well as the experiment related interface for accessing the websites while having the gaze position recorded. The additional show.html page designed for debugging consists of a single input field which expects a valid ID, which is then used to get the fixations from the database. Since it's a debugging page, it's not made to be used by all users and therefore it has very little precautionary features built in. As an example, an incorrect ID wouldn't return an error, it will just try to get the fixations with an incorrect ID from the database. The show.html page displaying fixations doesn't display the sequence of the fixations, which makes long sessions hard to visually analyse.

### 3.1. Website for the Study

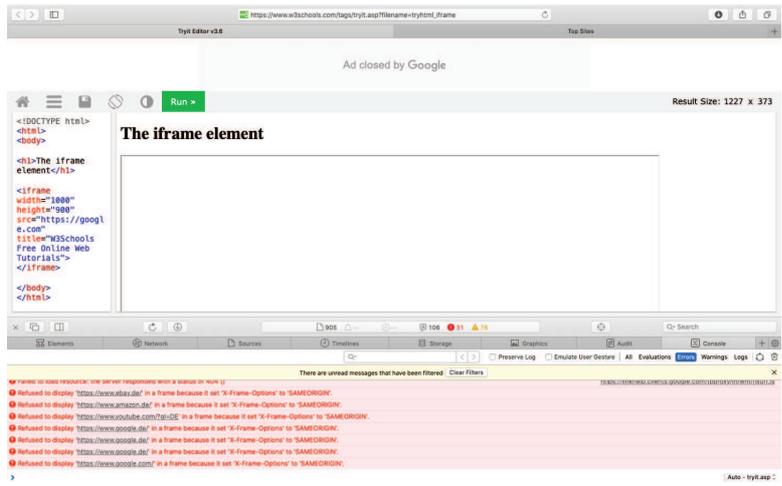


Figure 3.5: An example of an x-frame-origin not allowing usage of the website inside an iframe

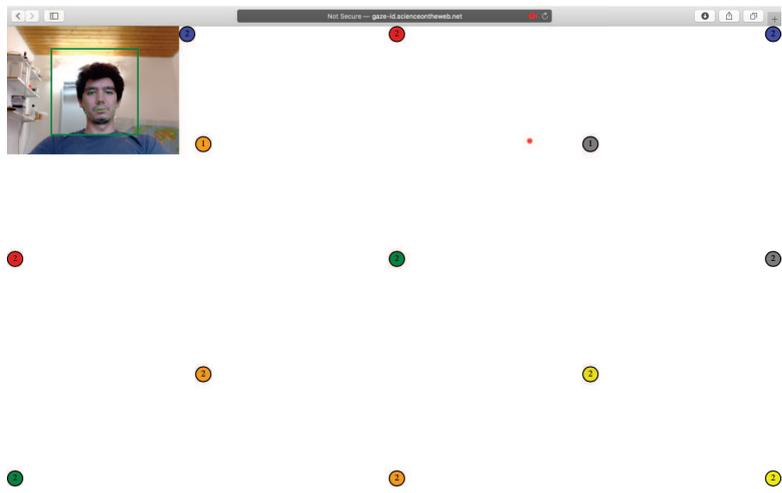


Figure 3.6: Calibration stage, the estimated gaze is shown as the red dot

The main html page has everything the participant in the experiment for this study will need during the whole experiment. The calibration stage of the experiment on this website is implemented as a pattern of small round clickable circles with different colours.

The round small clickable elements used for the calibration are simple HTML div [80] elements with applied CSS parameters shaping them round and painting them in different colours. All of the round small clickable elements used for calibration have onClick event handlers that decrease their value with every click. The value of each of those elements is important for the calibration process, since each element is expected to be clicked twice, and each time each element is clicked, the value of

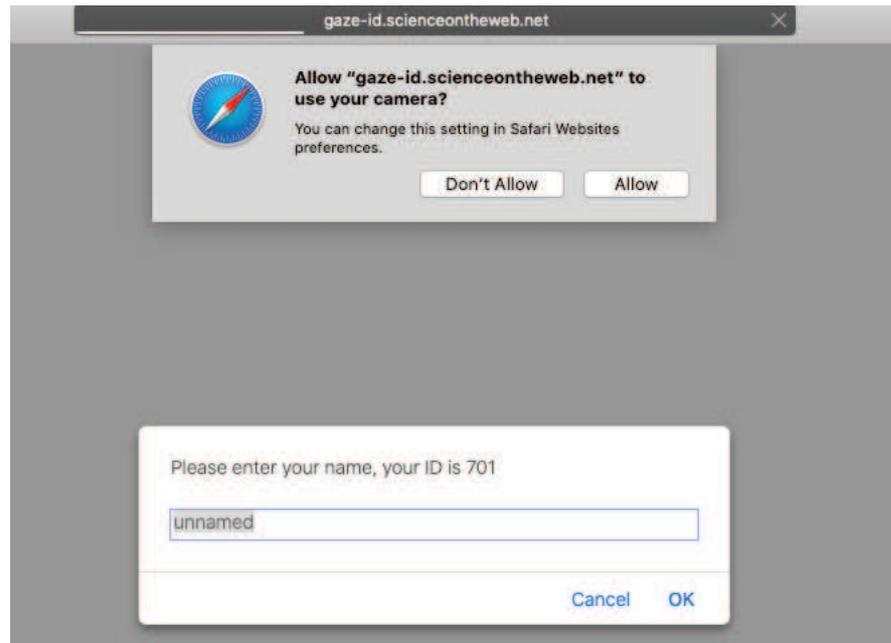


Figure 3.7: The request to use the webcam handled by the browser and the JS-alert window asking for a name while displaying the unique ID

the clicked element, which is also displayed on the element, is being decremented until it reaches 0, when the element stops being displayed. While the user is busy clicking the calibration elements, WebGazer is adding the corresponding eye feature samples collected during clicks and mouse moves to the model.

### 3.1.2 Functionality

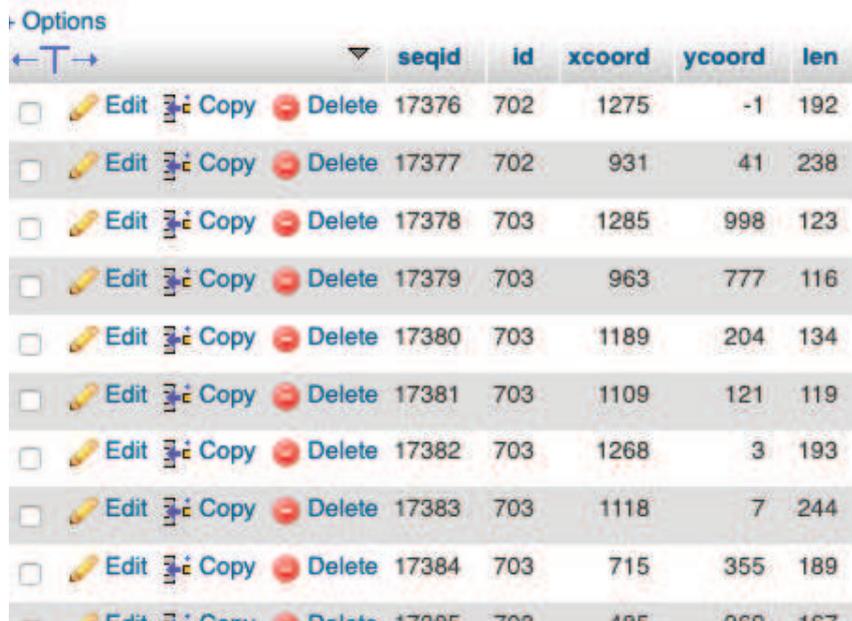
The website functionality can be divided into three consequent stages, between which the website switches automatically. The first stage is the stage with the JS-alert window asking for the name and displaying the ID. The second stage is the calibration stage and the third stage is the actual scanpath recording while the user browses one of the five offered websites. The first stage automatically changes to the second stage after the user completes the initial task of typing in a name, and the second stage switches into the third stage immediately after the user is finished with calibration. In the third stage, in which the user can browse five different websites, no further changes are made. The user is free to browse the websites as long as desired.

The first stage consists of the browser request for the webcam, which is handled by the browser entirely, and the JavaScript alert window displaying the user ID and asking for a name. During the first stage, not a lot of actions happen to the user, but most of the initialization process happens during the first stage. From the

### 3.1. Website for the Study

moment the website is loaded, the calibration is going on. The scanpath is not being saved to the database yet, but each click the user makes on the alert-window, every cursor movement is being added to the RR model used by WebGazer, even before the calibration screen is displayed.

The first thing the browser does is it assigns the user an ID. An ID is one of the most important elements of a recording because it is the only key in all tables with different information. Without the ID, it would be impossible to link a fingerprint or a name to a scanpath, which is vital for this experiment. The ID is present in every database entry in each table (Figure 3.8).



			seqid	id	xcoord	ycoord	len
<input type="checkbox"/>	Edit	Copy	Delete	17376	702	1275	-1 192
<input type="checkbox"/>	Edit	Copy	Delete	17377	702	931	41 238
<input type="checkbox"/>	Edit	Copy	Delete	17378	703	1285	998 123
<input type="checkbox"/>	Edit	Copy	Delete	17379	703	963	777 116
<input type="checkbox"/>	Edit	Copy	Delete	17380	703	1189	204 134
<input type="checkbox"/>	Edit	Copy	Delete	17381	703	1109	121 119
<input type="checkbox"/>	Edit	Copy	Delete	17382	703	1268	3 193
<input type="checkbox"/>	Edit	Copy	Delete	17383	703	1118	7 244
<input type="checkbox"/>	Edit	Copy	Delete	17384	703	715	355 189
<input type="checkbox"/>	Edit	Copy	Delete	17385	703	485	868 167

Figure 3.8: The user ID is the only key for the fixations and the fingerprints in this database. Without the key, it is impossible to get the correct fixations.

The ID is not a random number, it is the sequential number of the user taking part in the experiment. As an example, in Figure 3.8 we see the ID 702 and 703. Those IDs represent the 702nd and the 703rd session of eye scanpath recordings, and they are consequent, which is convenient enough to have easily usable numbers that are not longer than needed. The ID is being assigned by calling the JavaScript function "getNewID()", which tries to assign a proper ID to the user. The getNewID() function calls the id.php script with the time in ms since January 1, 1970 returned by Date.now(), as an argument. This number, when received by id.php, gets saved into the database, into the "userlist" table with an "auto\_increment" parameter. The table creates a new, incremented number for each database entry. The time in ms being saved in that table gets saved in a new entry with a new sequential ID number. The entry in the database is then requested by the time number, which returns the consecutive number of the entry in the database with the given timestamp value.

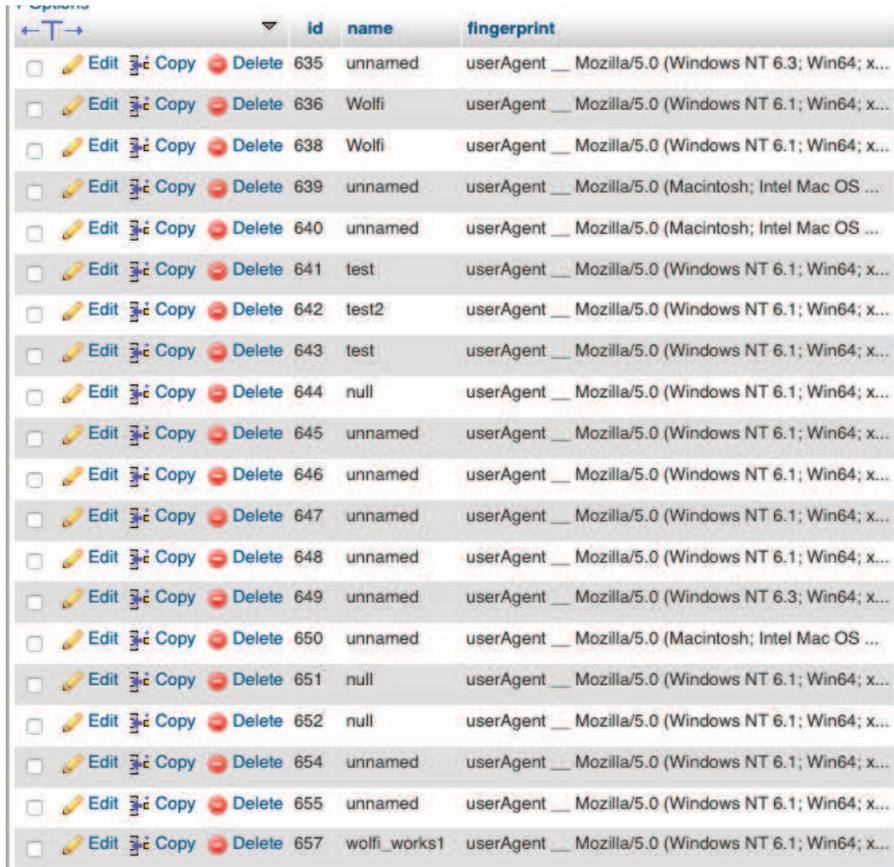
The consecutive entry number in the database is the ID for this session. The ID is then returned by `id.php` to the JavaScript `getNewID()` function, which saves it as an ID, which is being used in any database request. The process of getting the ID is made so complex in order to avoid the possibility of two users getting the same ID, if they open the webpage at the same time. The only scenario, in which two users would get the same ID in this current implementation, would be if two users would send the request for a new ID in the exact same millisecond, which is very unlikely given that this website is not even expected to be used by more than one user at the same time.

About 200 ms after the page has loaded, the browser fingerprint is being saved with the function `getBrowserFingerprint()`. The `getBrowserFingerprint()` function gets the browser fingerprint with an open-source fingerprint gathering tool, `FingerprintJS` [82]. This process usually doesn't require any user actions and by the time the user starts to type his name, or even manages to close the alert-window asking for a name, the fingerprint is saved. The user name is requested in a JavaScript alert-window, which is initiated by calling the function `getName()`. The function `getName()` shows a simple window asking for a user name and displaying a user ID. The username and the fingerprint are then saved to the database (Figure 3.9).

After the username has been submitted, the calibration process is being started (Figure 3.6). The initial part of the calibration is the arrangement of the small round calibration elements in a pattern on the user screen. A JavaScript function calls the small round div elements of the page DOM [83] and adjusts their location on the screen (Figure 3.10).

Each of the small round calibration objects is clickable and is expected to be clicked exactly twice. Each time the round clickable calibration component is being clicked, a JavaScript event handler calls a function to decrease the individual round clickable element click counter. When the click counter reaches 0, that particular clickable element with the counter at 0 disappears. Apart from a counter for each of the elements, there is a global counter for the whole calibration stage which counts how many times each round clickable element has been clicked. Since there are 13 round clickable elements, the global calibration counter expects each element to be clicked twice – 26 clicks on the round small objects in total. Each time each round small clickable object is clicked, the global counter is changed too. The moment the global counter reaches 0, the calibration stage immediately switches to the gaze recording stage and changes the `calibration_complete` variable to true, to inform the function handling the gaze recording that the calibration is finished, which would allow the recording function to start saving the estimated gaze points to the database. Each of the small round clickable objects has a colour and a number inside, visible to the user performing the calibration. The colour is randomly chosen, and the number represents the number of times the user has to click on that particular round clickable object until that particular small round clickable object disappears. Since there is one global counter for all clicks on all round clickable object, in case one

### 3.1. Website for the Study



			id	name	fingerprint
<input type="checkbox"/>	Edit	Copy	635	unnamed	userAgent __ Mozilla/5.0 (Windows NT 6.3; Win64; x...
<input type="checkbox"/>	Edit	Copy	636	Wolfi	userAgent __ Mozilla/5.0 (Windows NT 6.1; Win64; x...
<input type="checkbox"/>	Edit	Copy	638	Wolfi	userAgent __ Mozilla/5.0 (Windows NT 6.1; Win64; x...
<input type="checkbox"/>	Edit	Copy	639	unnamed	userAgent __ Mozilla/5.0 (Macintosh; Intel Mac OS ....
<input type="checkbox"/>	Edit	Copy	640	unnamed	userAgent __ Mozilla/5.0 (Macintosh; Intel Mac OS ...
<input type="checkbox"/>	Edit	Copy	641	test	userAgent __ Mozilla/5.0 (Windows NT 6.1; Win64; x...
<input type="checkbox"/>	Edit	Copy	642	test2	userAgent __ Mozilla/5.0 (Windows NT 6.1; Win64; x...
<input type="checkbox"/>	Edit	Copy	643	test	userAgent __ Mozilla/5.0 (Windows NT 6.1; Win64; x...
<input type="checkbox"/>	Edit	Copy	644	null	userAgent __ Mozilla/5.0 (Windows NT 6.1; Win64; x...
<input type="checkbox"/>	Edit	Copy	645	unnamed	userAgent __ Mozilla/5.0 (Windows NT 6.1; Win64; x...
<input type="checkbox"/>	Edit	Copy	646	unnamed	userAgent __ Mozilla/5.0 (Windows NT 6.1; Win64; x...
<input type="checkbox"/>	Edit	Copy	647	unnamed	userAgent __ Mozilla/5.0 (Windows NT 6.1; Win64; x...
<input type="checkbox"/>	Edit	Copy	648	unnamed	userAgent __ Mozilla/5.0 (Windows NT 6.1; Win64; x...
<input type="checkbox"/>	Edit	Copy	649	unnamed	userAgent __ Mozilla/5.0 (Windows NT 6.3; Win64; x...
<input type="checkbox"/>	Edit	Copy	650	unnamed	userAgent __ Mozilla/5.0 (Macintosh; Intel Mac OS ...
<input type="checkbox"/>	Edit	Copy	651	null	userAgent __ Mozilla/5.0 (Windows NT 6.1; Win64; x...
<input type="checkbox"/>	Edit	Copy	652	null	userAgent __ Mozilla/5.0 (Windows NT 6.1; Win64; x...
<input type="checkbox"/>	Edit	Copy	654	unnamed	userAgent __ Mozilla/5.0 (Windows NT 6.1; Win64; x...
<input type="checkbox"/>	Edit	Copy	655	unnamed	userAgent __ Mozilla/5.0 (Windows NT 6.1; Win64; x...
<input type="checkbox"/>	Edit	Copy	657	wolfi_works1	userAgent __ Mozilla/5.0 (Windows NT 6.1; Win64; x...

Figure 3.9: The database with the user names and their browser fingerprints. Users rarely bother to use an actual name, which is why an ID is used everywhere.



Figure 3.10: The small round object with a number on it is used for calibration

of the small round objects gets clicked more than twice, it will change the counter value by 3, which might lead to a change of the calibration mode to the recording mode without the calibration being properly done. This has not happened so far, since the JavaScript function counting the clicks is faster than the speed of a human double-clicking.

After the calibration counter reaches 0, the iframe with the first of the five websites changes its HTML "hidden" attribute [84] from true to false, and the hidden iframe

## Chapter 3. Method

becomes visible to the user. Five interface buttons on the sides of the screen used to change displayed website become visible too. By clicking each of the interface buttons, an event handler triggered by a mouse click on those buttons gets called and replaces the website address in the iframe.

The debug tool, `show.html`, has been implemented to show all fixations for a certain user ID. `show.html` has a single input field with a button to submit the ID number typed in the input field. After submission, the ID number is being sent to `getFixations.php`, which retrieves all fixations from the database by that ID. The fixations are then shown as squares on the screen.

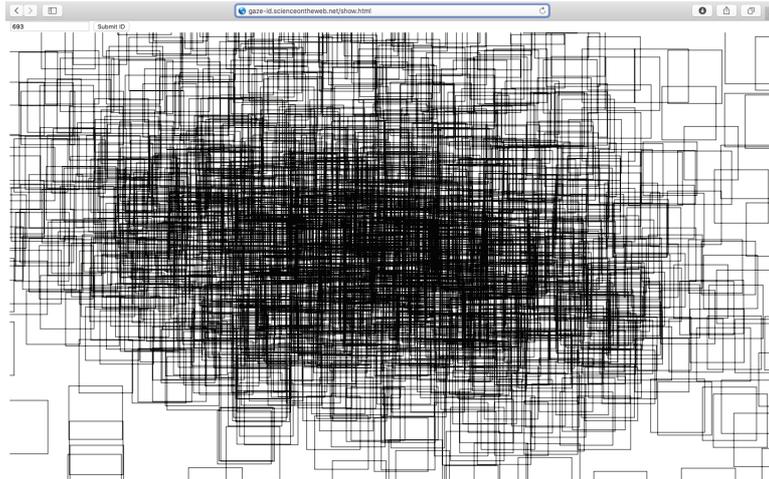


Figure 3.11: The debugging tool used to visualize fixations for a certain ID from the database, `show.html`. The fixations for ID 693 are shown.

WebGazer is active from the moment the page is loaded and the ridge regression model used by WebGazer is constantly being updated. Gaze predictions are being offered to the browser from the very first clicks the user makes too, even before calibration starts. The predictions are taken by the browser, but they are not saved into the fixations database until the calibration is complete. The handling of the gaze prediction with WebGazer, which is a JavaScript module, is processed by a callback-function – a function which is being called by the module each time there is a new prediction available. The callback function gets the prediction point as a pair of coordinates. The pair of coordinates is given to the `detectFixations()` function, which detects fixations. WebGazer constantly offers predictions of the gaze, even if the gaze is affixed. The relevant information that needs to be saved into the database are the fixations. Saccades are detected as changes in fixations. In order to filter false saccades reported by WebGazer, a detection radius is used for each fixation, within which every prediction is considered to be the same fixation and is not being detected as a saccade. The detection radius is set to 100 pixels. After a fixation is detected, all successive gaze estimations within 100 pixels are ignored and assumed to be the same fixation. In addition to a detection radius, another filtration technique

is used to filter out possible false estimations. In order for a fixation change to be detected, the fixation change has to happen at least 100ms after the previous fixation change. Fixation changes within 100ms are usually false estimations. When a fixation change is detected and not removed by a filter, the fixation gets given to the function `saveToDB()` as an argument. The function `saveToDB()` sends the fixation to a php script `savefixation.php` if the calibration is completed, and if the calibration is not completed, the `saveToDB()` function doesn't send the fixation to the php script. The php script `savefixation.php` saves the fixation to the MySQL database, into the table with fixations, which is named "fixations", using the user ID as a key. All fixations can be accessed in the table. Multiple fixations can be saved each second.

Since the fixations are sent to the database immediately after each fixation is detected, there is no distinct or required method of "finishing" the experiment. The experiment ends when the browser window is closed, or when the internet connection between the user and the server disappears. The fixations are not being lost after the end of the experiment since they already are saved on the server.

#### 3.1.3 Process

The website starts in the first stage with a request to use the user's web camera, if the browser settings require such a request to be made (Figure 3.8). Since the camera use request is being handled by the browser, it has no influence on the functionality of the website besides of the website being temporarily covered by that request. The website welcomes the user with a JS-alert window with a text input field asking for the user's name while displaying the user's unique ID number (Figure 3.8), which can be used to relate to that particular recording session and to request the fixations through the debugging tool `show.html`.

After typing in a name, or leaving the field unchanged and submitting it, the JS-alert window disappears and the calibration stage is being shown, with the calibration pattern – differently coloured small round clickable objects with numbers displayed on a blank page with the webcam video on the top left corner, with the face detection outline layed on. During calibration, the user is expected to click each small round object 2 times. The number on the small round object represents the number of times that object still has to be clicked (Figure 3.10).

While the user clicks on the calibration pattern, a small red dot representing the current estimated gaze position is displayed on the calibration screen. After the user is finished clicking all elements of the calibration pattern enough times, the calibration screen disappears, and the actual recording interface is being shown to the user.

In the recording interface, the user is immediately taken to one of the websites, with the gaze recording running, with five small buttons in the corners and in the middle of the upper side of the screen available to the user to switch between users (Figure

## Chapter 3. Method

3.12).

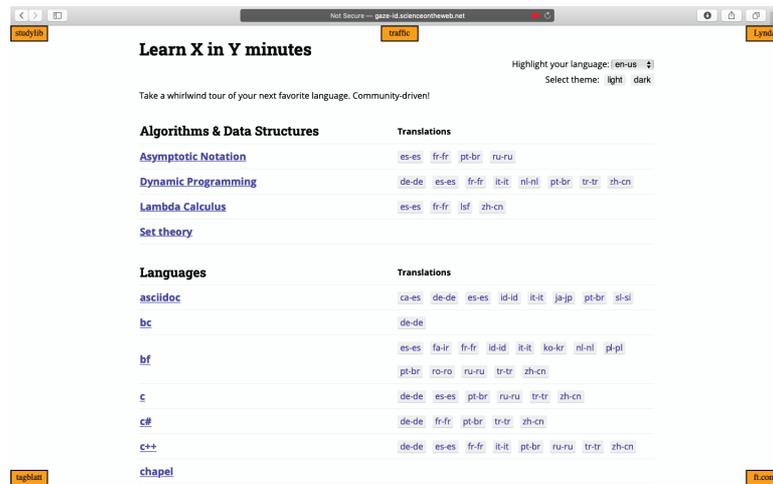


Figure 3.12: The gaze recording stage, with five websites available with five orange buttons in the corners

Each time the user clicks on one of the orange buttons in the corners, the website being displayed to the user changes. The user is free to explore, browse and even click on links on the website. There are no built-in limits to this process and theoretically the recording can take forever. Practically, the user will get tired, the database might overflow or the browser will stop properly working. The last stage stops when the user closes the website by closing the tab/window/browser.

## 3.2 Data Gathering

The process of gathering gaze position data is made as simple for the end user as possible, in order to avoid overstressing the user being experimented on. It is important that the user feels comfortable and doesn't deviate from his usual behaviour while browsing the website. The initial stages of the experiment are made as simple as possible and are made to be completed as fast as possible, in order to spend as much time recording the browsing eye movements.

### 3.2.1 Process

In this experiment, two individuals have been experimented on. The experiments on each of the two individuals have been conducted on different days, with one person using two different systems. There has been a total 20 sessions of data gathering, 10 sessions for each user. During each session the user has been browsing the provided websites for several minutes, with the websites being changed between freely.

During each session, the user has been placed in front of the web camera, in a position which would allow web gazer to correctly estimate the position of the user face and record it. For each session, a separate calibration procedure has been performed. During each session, the users have been browsing the correctly displayed 5 websites, which they were allowed to switch between as often as they want. The experiment was finished when the users would feel like they have browsed for long enough.

### 3.2.2 Grid

The user gaze fixations are distributed across their screens, which have variable resolution and the gaze estimation accuracy varies highly across different sessions based on different calibration procedures. In order to normalize the fixation positions relative to their screens, each screen has been divided into a 5x8 grid. This grid rate has been chosen in order to allow for almost perfectly square grid cells for most of modern screens, including all screens used by the participants, which have a 10:16 aspect ratio. The gaze positions, when divided into a 5x8 grid, distributed across 40 cells, are normalized relative to the screen and can be compared even with different screen resolutions. A larger cell count has been considered, but a 5x8 grid perfectly satisfies the estimation accuracy provided by WebGazer.

### **3.3 Evaluation with MatLab**

The collected data from the experiment has been processed and evaluated using MatLab. The datasets were exported from the database into comma separated value files that are easily readable by MatLab.

#### **3.3.1 Process**

In order to evaluate the Data, first, the necessary data has been imported to MatLab from the comma separated value files with the data. In order to import the browser fingerprint data, which is represented as text, each of the 29 browser parameters has been encoded into a numerical representation, in order for the decision tree building algorithm to be capable of taking the values as parameters. The fixations of each session are divided into sections of a 8x5 grid, with each grid cell having the duration of all fixations in that grid cell. The values in the grid are then normalized to represent the relative percentage of time a fixation being active. In order to compare the improvement in browser fingerprinting, a dataset with both the scanpath data and the browser information is created, which can be used in our classification algorithm for classification of the combined information about the user from browser fingerprinting and from eye tracking.

## 4 Evaluation

The evaluation of the results of our study is the most important part of this study and therefore has been being prepared for from the first steps into this study. For the evaluation, experimental data from 2 individuals has been collected. The experimental data contains the browser fingerprint and the eye tracking data. The experimental data has been adjusted in order to be easily processed by various decision tree building algorithms, as well as combined in order to detect any improvements. The aim of this study is to research the possible improvement eye tracking could make to the existing browser fingerprinting techniques.

In order to evaluate the combined fingerprinting and eye tracking performance, we will first evaluate the performance of the conventional browser fingerprinting technique, after which we will evaluate the performance of our eye tracking technique, after which we will analyse them combined to see any evidence of improved browser watermarking performance. All these evaluations will be conducted using different cross-validation algorithms, because we only have two individuals that have been experimented on, and different cross-validation algorithms might offer different fingerprinting accuracy. All evaluation is going to be performed in the MatLab Machine Learning Toolbox.

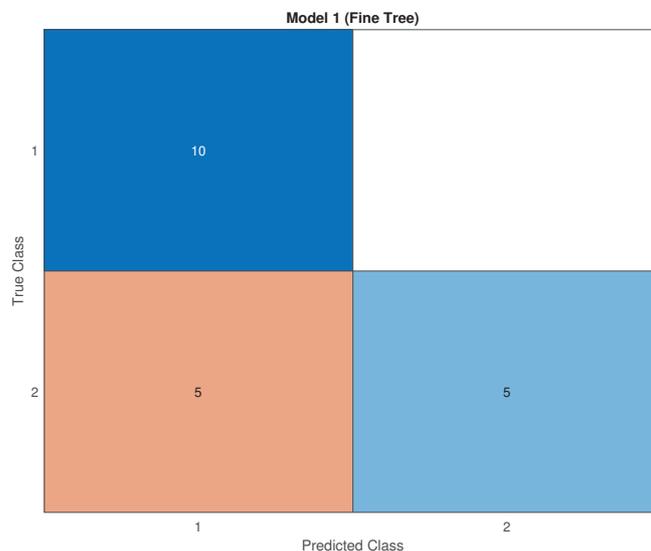


Figure 4.1: The confusion matrix and the accuracy of the Fine Decision Tree algorithm for browser data

## Chapter 4. Evaluation

The first algorithm that is going to be used will be the Fine Decision Tree algorithm, which should offer the performance of a normal decision tree. The Fine Tree algorithm grows a classification tree with many leaves and tries to find many distinctions between classes, with the maximum number of splits being 100. Fingerprinting based on browser data is being evaluated first and as shown on Figure 4.1, the evaluation results of pure browser fingerprinting offer 75% accuracy, but because of different browser combination used by the two test subjects, the accuracy is different for the test subjects. For the test subject 1 all samples were correctly classified, because test subject 1 has been using only 1 browser. For test subject 2, only 50% of the predictions were correct, but test subject 2 has been experimented on with different browsers, including the same browser test subject 1 has been using, which makes an accurate cross-validation based on browser fingerprint data fairly biased.

Next, the eye tracking data is being evaluated for Fingerprinting on Figure 4.2.

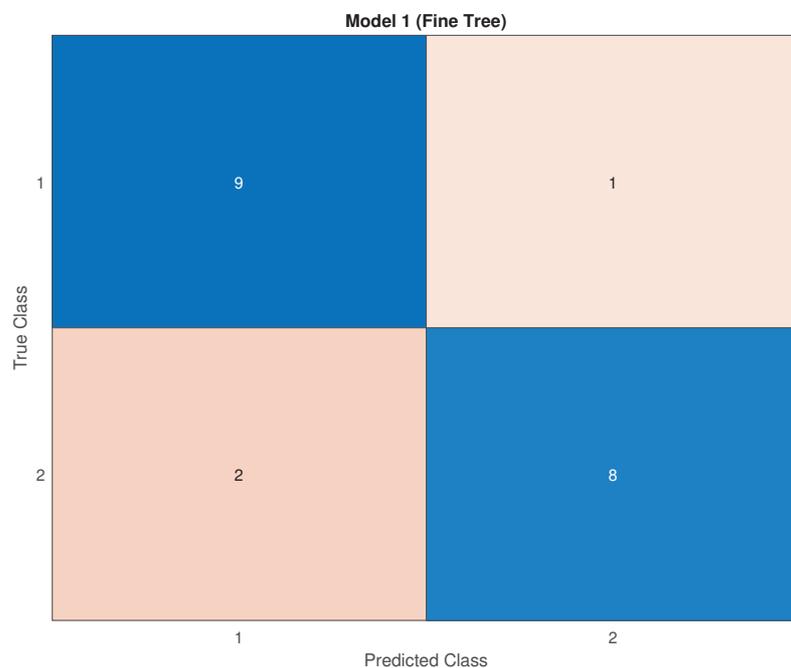


Figure 4.2: The confusion matrix and the accuracy of the Fine Decision Tree algorithm for eye tracking data

The pure eye tracking data fingerprinting analysis has performed much better than the browser data fingerprinting using the fine decision tree cross-validation algorithm. An accuracy of 85% could be considered fairly high, with the accuracy for test subject 1 being at 90% and the accuracy for test subject 2 being at 80%. Comparing these results, eye tracking data has been much better at being classified properly for test subject 2, and for test subject 1 the eye tracking data has been almost as precise as browser data fingerprinting.

Now that the cross-validation results for both pure eye tracking data and pure browser data classification using Fine decision trees are available, the evaluation of the combined data used for classification is being shown on Figure 4.3.

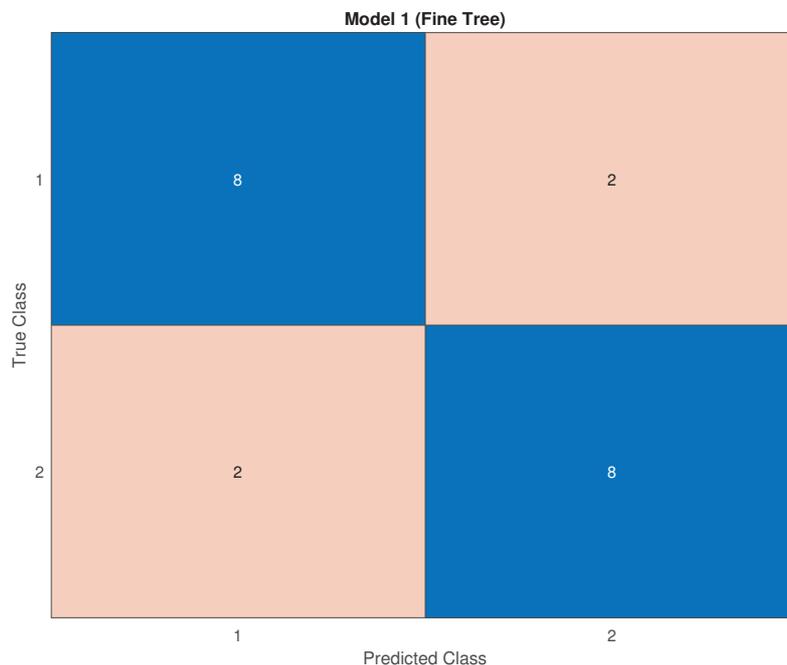


Figure 4.3: The confusion matrix and the accuracy of the Fine Decision Tree algorithm for combined eye tracking and browser data

The combined eye tracking and browser fingerprinting cross-validation results offer 80% accuracy for both test subjects, which is higher than then the browser-based classification accuracy for test subject 2, and the accuracy for test subject 1 is close enough to the initial browser classification accuracy. Although the different proportions of the number of browsers used by both test subjects introduces a lot of bias to our evaluation, specially for test subject 1, the accuracy for test subject 2 can actually be used as a proper evaluation score, and in this case, using the fine tree cross-validation algorithm, the fingerprinting accuracy for test subject 2 has been lifted from 50% to 80%, and the overall testing accuracy has been lifted from 75% to 80% which is, despite being small, an actual improvement that has been measured. Interesting is the fact that the overall accuracy for the combined data classification has been reduced compared to the pure eye tracking data, but increased compared to the pure browser information. It might seem as if the pure eye tracking browser fingerprinting technique is more accurate, but this inaccuracy is probably caused by the small sample size consisting of just 20 samples for each data type, 10 for each of the 2 individuals being experimented on.

In general, using the fine decision tree cross-validation algorithm, eye tracking

## Chapter 4. Evaluation

combined with conventional browser fingerprinting data does improve the overall classification accuracy in this study. Although using pure eye tracking data without browser fingerprinting data combined with the fine tree classification algorithm has performed overall better.

The results for fine trees have shown some improvement, but very slight. Fine trees try to classify samples using many fine distinctions. In case our experimental data is tree-type sensitive, the next algorithm to be used will be the Coarse Decision Tree algorithm, which is going to have up to 4 splits, compared to 100 splits used in fine trees. As usual, in order to evaluate the improvement in classification performance for combining eye tracking data with browser data, first, classification has to be performed based on pure browser data, as shown on Figure 4.4.

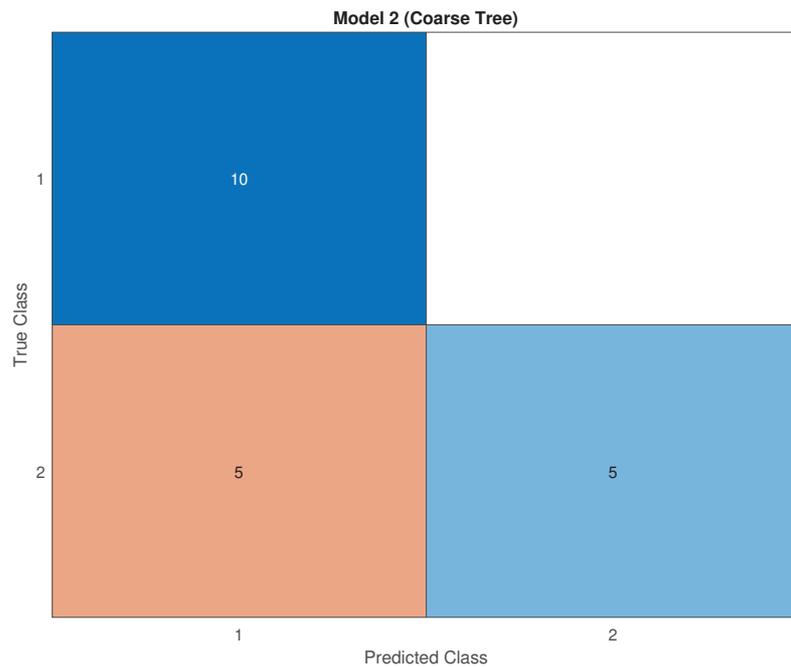


Figure 4.4: The confusion matrix and accuracy for a Coarse Decision Tree algorithm used on pure browser data

The coarse decision tree algorithm used for the experimental data cross-validation has performed with a 75% accuracy on pure browser data, which is similar to the performance of the fine decision tree algorithm with the same distribution as the distribution of the accuracy in the fine decision tree algorithm, which is the result of the experimental data being biased based on the different number of browsers used by different experiment participants. Similar to the results of the fine tree algorithm, the accuracy for test subject 1 is 100%, which is caused by the fact that test subject used a smaller number of browsers test subject 2 used, which is why the accuracy for test subject 2 is 50%, however, the overall accuracy is 75%, and this accuracy

offers a great baseline for classification performance comparison.

In order to properly compare the classification results using the combination of browser and eye tracking data, the coarse tree cross-validation algorithm first has to be used on pure eye tracking values, which would make a deeper analysis possible. The coarse tree algorithm classification results on the experimental data are shown on Figure 4.5.

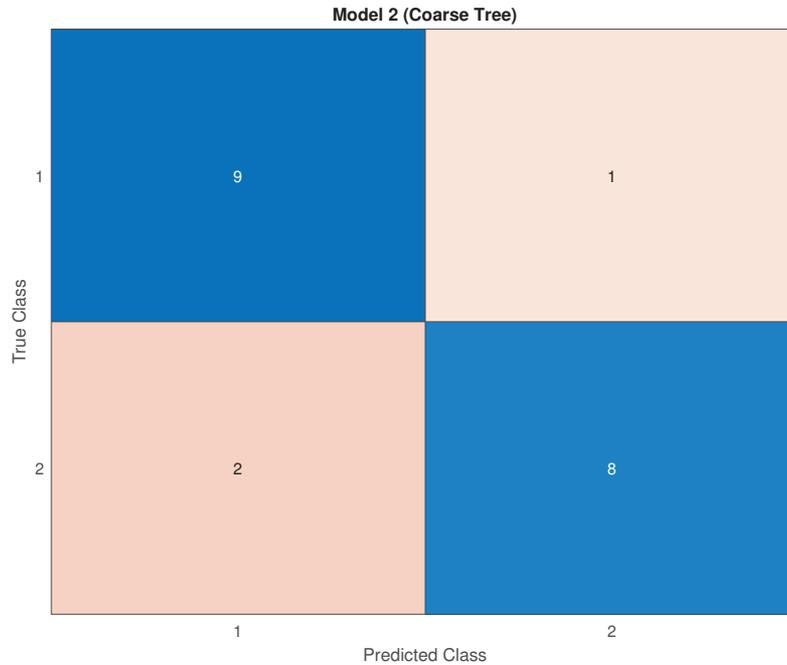


Figure 4.5: The confusion matrix and accuracy for a Coarse Decision Tree algorithm used on pure eye tracking data

An overall accuracy of 85% provided by the coarse decision tree cross-validation algorithm based on pure eye tracking data is a great result that is very similar to the result of applying the fine tree algorithm to the pure eye tracking test data. In this result, the accuracy for each test subject does not seem to be biased and the accuracy for test subject 1 is at 90%, while the accuracy for test subject 2 is at 80%. The overall accuracy of classification using coarse decision tree for pure eye tracking data is much higher than the pure browser data results accuracy, and the accuracy is distributed with less variance across the test subjects – test subject 1 has 90% accuracy, while test subject 2 has 80% accuracy, with just 10% difference in their results accuracy, compared to 50% difference in the pure browser data accuracy for the coarse decision tree.

Having the accuracy for the pure eye tracking and pure browser data, it is possible to properly analyse the improvement in accuracy achieved by using both eye tracking

## Chapter 4. Evaluation

and browser data combined. When both the eye and browser experimental data are combined, a higher accuracy is possible if both datasets have been offering an accuracy higher than 50%, which has been the case. The results for the combined eye and browser data is shown on Figure 4.6.

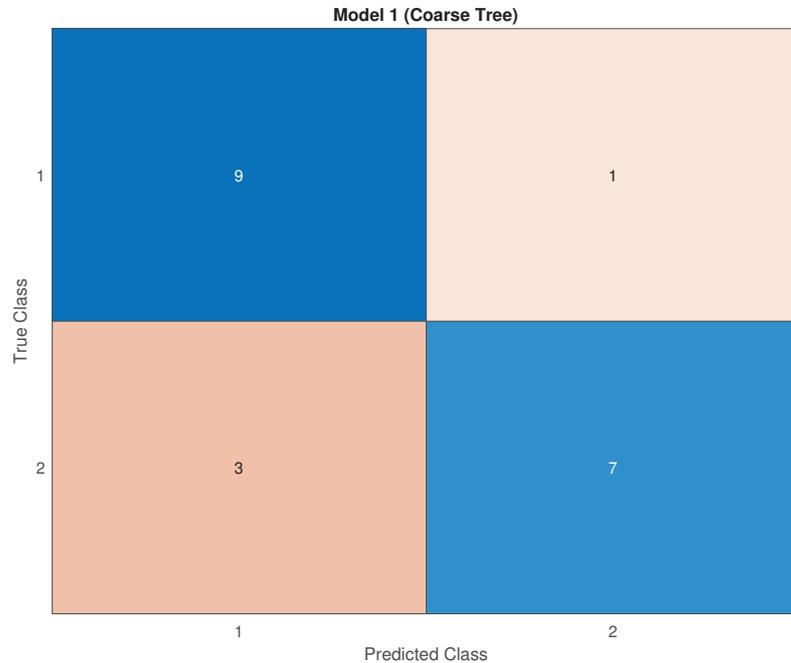


Figure 4.6: The confusion matrix and accuracy for a Coarse Decision Tree algorithm used on both eye tracking and browser data

In case of using a coarse tree, the overall accuracy for the combination of eye tracking and browser data is similar to the overall accuracy for the same dataset used in a fine tree, however, interesting is the fact that the variance of results for different test subjects is higher than the result variance in fine trees. In case of fine trees, both test subjects had an equal accuracy of 80%, which makes the variance 0%, however, in this part, coarse trees have shown an accuracy difference between the two test subjects at 20%, which is fairly high and close to the result variance observed in both fine and coarse trees for the pure browser data. In this case, the biased experimental browser data of test subject 1 might have had a greater influence on the results, because the accuracy for test subject 1 is 90% while the accuracy for test subject 2 is 70%. But the overall accuracy of combined data used with the coarse decision tree algorithm is the same as the overall accuracy achieved with the fine tree algorithm.

In general, coarse decision tree cross-validation offered similar overall accuracy results to the fine decision tree algorithm, with a slight deviation in accuracy variance among test subjects.

Now that different decision tree classification algorithms have been evaluated in use on the dataset, in order to try to improve the test results, a decision tree ensemble [Wik20k] is going to be used to classify our test samples. Decision trees are famous for suffering from high result variance, which has been observed even in our experiments based on a rather small sample size. In the case of this study, the training data consists of two parts and the previous evaluations have shown a high difference in accuracy and accuracy distribution across test subjects based on the combination used, which might be reduced in the following evaluation procedures using a method that would provide more stable results. A decision tree ensemble, which is sometimes called a bagged tree or bootstrap aggregated tree [Wik20e] creates multiple trees based on random subsamples taken from the dataset. Tree bagging reduces variance in results, which has been fairly high in our results. Apart from bagged tree ensembles, there are boosted trees and random forest trees, but these ensembles are non-relevant.

Now that it's time to perform the experiment using tree ensembles, the results might differ. In this part of the experiment, a bagged tree is going to be used, since a bagged tree should offer the best variance reduction. Just as in the previous parts of the experiment, in order to properly evaluate, how a dataset improves classification, it is important to have the baseline results, which is why the first bagged decision tree used for cross-validation will be used on the pure browser fingerprint data (Figure 4.7).

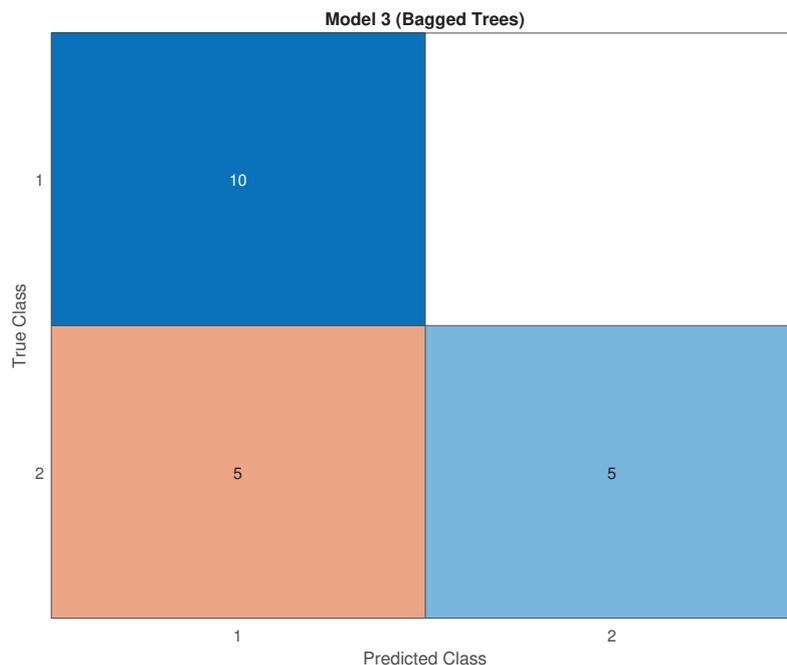


Figure 4.7: The confusion matrix and accuracy for a Bagged Tree algorithm used with browser data

## Chapter 4. Evaluation

Despite using a bagged tree, the pure browser fingerprinting dataset classification accuracy is similar to the accuracy that has been achieved with previous algorithms, which is probably caused by the biased browser fingerprint dataset. The overall accuracy is 75%, but the variance between different test subjects is extremely high, with test subject 1 being classified with a 100% accuracy while test subject 2 is being classified with a 50% accuracy.

Using the bagged tree ensemble classification methodology on the eye tracking dataset is not expected to introduce a lot of change to our accuracy results, since the eye tracking dataset accuracy has had pretty low variance when used with single tree algorithms, and since the bagged tree is expected to reduce variance in high variance cases, the variance for this dataset has been pretty consistent and low. The bagged tree cross-validation results for the pure eye-tracking data are shown on Figure 4.8.

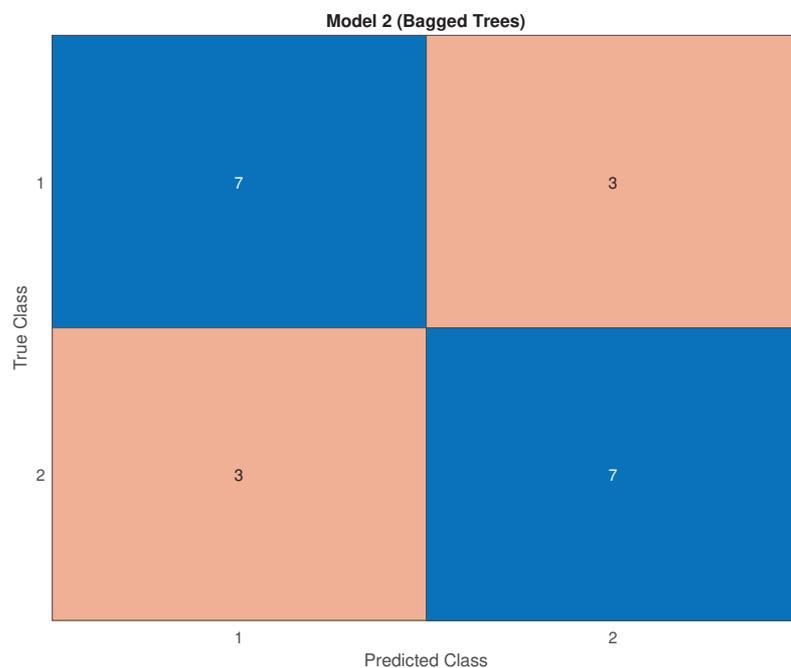


Figure 4.8: The confusion matrix and accuracy for a Bagged Tree algorithm for pure eye tracking data

Surprisingly, the overall accuracy for the pure eye data has dropped compared to the single tree algorithms when used as an input to the bagged tree. The overall accuracy achieved with a bagged tree is 70%, which is even lower than the accuracy achieved with the pure browser data, however, the accuracy variance among different test subjects is 0%, both test subjects have been classified with the same accuracy as the overall accuracy, 70%. The reason, why the overall accuracy has dropped with the bagged tree algorithm, is a great topic for future work.

Now that the datasets have been evaluated with the bagged tree classification algorithm, the datasets, when combined, are expected to produce higher accuracy compared to the single-tree algorithms used earlier, since the two datasets being combined have different variances that are probably affecting the resulting classification accuracy with single trees.

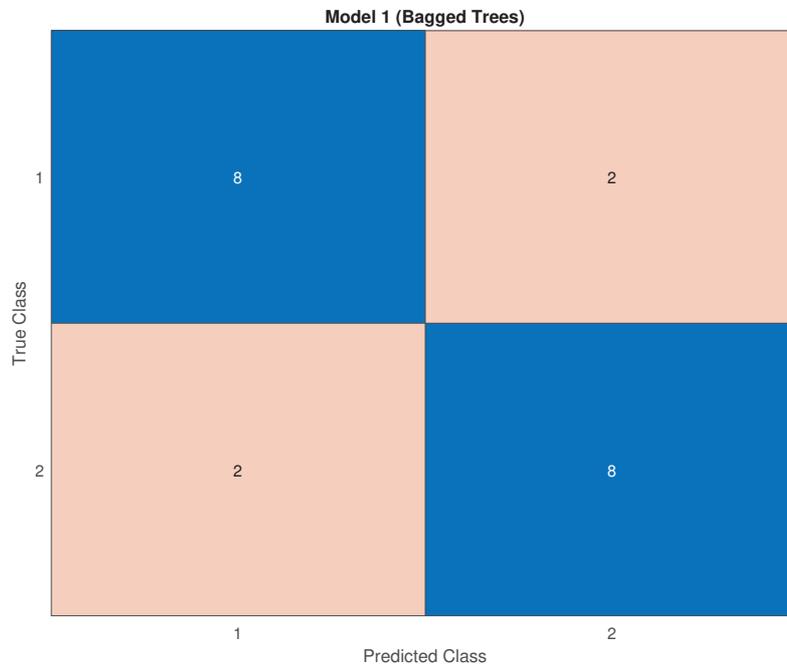


Figure 4.9: The confusion matrix and accuracy for a Bagged Tree algorithm for combined data

Surprisingly, the results of the bagged tree algorithm applied to the combined dataset show a 80% classification accuracy, which is slightly higher than the results of each of the pure datasets processed by the bagged tree classifier, but the overall accuracy is not higher than the classification accuracies achieved with single-tree algorithms. Despite not achieving higher accuracy by using a bagged tree compared to single-tree algorithms, while using a bagged tree, the accuracy of combined eye tracking and browser fingerprinting data has been improved compared to the accuracy of the separate datasets, which is what this study is expected to show. The overall accuracy gain with bagged trees is low, however, combining data as an input to a bagged tree reduces variance while still improving accuracy.

Overall, three different decision tree classification algorithms have been used to evaluate the dataset. In order to evaluate the dataset, each algorithm has been used to evaluate the eye tracking data and the browser data separately, and combined. The evaluation using fine trees offered 75% accuracy for pure browser data, 85% accuracy for pure eye tracking data and 80% for the combined data. Evaluation

## Chapter 4. Evaluation

using coarse trees offered 75% accuracy for pure browser data, 85% accuracy for for pure eye tracking data and 80% accuracy for the combined data. Bagged trees offered 75% accuracy for pure browser data, 70% for pure eye tracking data and 80% accuracy for combined data. With all algorithms, the pure browser data has been improved using eye tracking data, however, pure eye tracking data has been offering higher accuracy than combined data for both fine and coarse tree algorithms than combined data, the combined data has been offering higher accuracy for bagged trees, but the offered accuracy was still lower than the pure eye tracking accuracy achieved with fine and coarse tree algorithms.

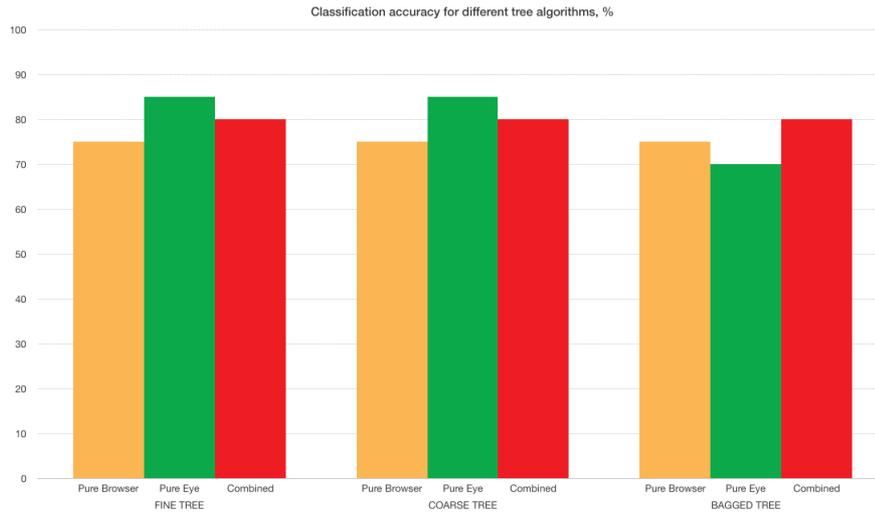


Figure 4.10: Classification accuracy based on the data types across different algorithms.

## 5 Discussion

This study is aimed at improving browser fingerprinting using a scanpath, a path followed by the user's eyes while using the web service. Although a scanpath is a biometric feature of humans, and doesn't quite fit into the definition of "system fingerprinting", we extract it through a JavaScript application running entirely in the browser of the end-user, which makes it a feature provided by the browser, allowing it to be used as part of a "browser watermarking" procedure, even though it's questionable whether the data is provided about the system and browser, or about the human. From a technical point of view, it's another set of datapoints provided by a function which could be called from a browser, which makes it suitable to be involved in the actual browser watermarking procedure.

Various decision tree algorithms have been used in this study, and the data has been prepared to be easily processed by these machine learning algorithms, which is why all the experimental data has been adjusted to be in a numeric format. The data being in a numeric format made the combined usage of the scanpath and browser data much easier.

The pure browser watermarking results in this study have been pretty inaccurate, which is caused by the relatively small number of participants in the experiment, as well as the relatively small number of different browsers used. For this study, the dataset quality for browser watermarking is not relevant, since this study is not researching browser watermarking accuracy and the browser accuracy provides just a baseline for comparison.

This study is aimed at researching the possible improvement introduced to the fingerprinting technology by eye tracking data retrieved from a normal webcam. Human verification and authentication based on eye movements is a relatively new subject in Computer Science, since eye tracking is a relatively new technology made accessible to the general public by the improvements in digital image sensors, capable of providing accurate eye tracking. Devices, designed specially for eye tracking offer an incredible accuracy, but even built-in webcams and relatively cheap image sensors offer acceptable accuracy nowadays, which makes this incredible new safety level possible for a wide range of applications. Eye tracking data, basically being an additional set of datapoints representing the user, without doubt is going to improve the accuracy when used in machine learning classification algorithms, which this study has successfully shown based on different algorithms, including single decision trees and tree ensembles such as bagged trees. The quality of modern

## Chapter 5. Discussion

web cameras is perfectly suitable for eye tracking with good enough precision, since in most use cases of web cameras user eyes need to be trackable too – eye movement tracking is an essential part of a conversation, even if it's a video call conversation.

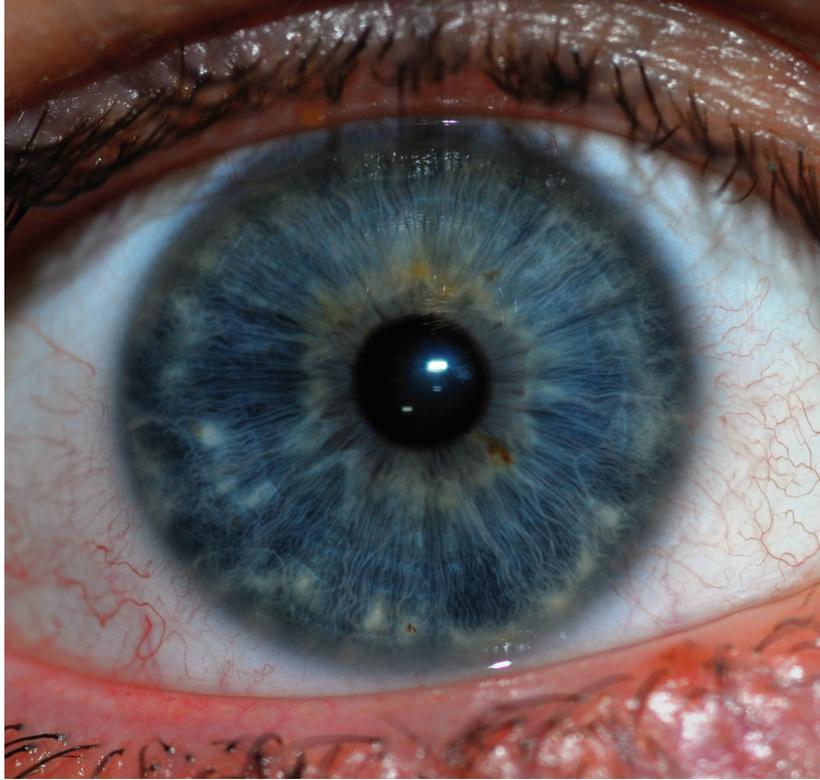


Figure 5.1: A high-resolution image of a human iris with the human eye around the iris [Wik20p]. This biometric feature cannot be captured by modern technology available in every laptop, opposed to the eye movements

The reality of eye tracking is, that eye tracking data actually is a biometric feature of a human that can be accessed through a browser. It is very similar to a proper fingerprint of a human or an iris image.

Eye tracking data, unlike most of other human biometrics, is accessible through any modern web camera and even phone camera, which are used in most modern devices because of the recent popularity of video calls and overall different applications. Browsers were made capable of handling web camera data due to the incredible entertainment opportunities web camera footage in browsers offer, and although the main purpose of web cameras is entertainment, their high enough image quality allows for them to be used as a tool to extract one of the few biometric features available to a remote server through a simple browser.

Eye tracking data is not the only biometric data available to a computer, many

computers have fingerprint readers, some of the fingerprint readers even have advanced biological metrics used for authentication, but there is no simple method of sending or requesting fingerprint data through a browser, since fingerprints cannot be used for entertainment. There are many more biometrics used by browsers, such as voice or even mouse movements, but a human doesn't constantly produce voice while browsing. Eye movements happen constantly, even when the mouse is not being moved. Eye movements seem to be the optimal biometric feature available through a browser.

Due to reasonable privacy concerns, all browsers don't allow just any website to process web camera footage. Even though WebGazer [PSL<sup>+</sup>16b] never sends or saves any images, it is reasonably complicated for the browser to inspect any script handling browser data, which is why browsers let the users decide, whether or not access to the web camera should be granted. Since granting access to web camera for most users is something that is only required when actual web camera usage for entertainment is involved, access to the web camera without any apparent reason for the user is suspicious. The suspicions users have are relatable when asked for web camera permissions – if there is no obvious reason to use the web camera, what is the purpose of using the web camera? And indeed, a web camera offers access to biometrics beyond the system, device and even location. Biometric data is pretty unique and is a feature of the actual human, and probably should not be used in the same fashion browser fingerprinting is used. Browser and system fingerprinting is used by all large corporations nowadays, and the browser fingerprint features represent the system, the device, even the location, but not the actual human, and it might be reasonable to not require an actual human biometric feature extraction for low-risk tasks like targeted advertising and access control to entertainment websites. However, it is always desired to improve the accuracy of browser fingerprinting, sometimes even for user convenience, which is why any information available to the service provider, be it a simple browser user-agent, an IP address or even eye movement data will be used in order to improve the service quality, which only benefits the user most of the times. Still, it is highly unlikely for any user to grant web camera access when involved in low-risk tasks without the direct involvement of a web camera, which leaves only a narrow field for browser-based eye tracking with modern technology consisting of mostly banks and other high-risk applications where a user is ready to give up some privacy in return for extra safety.

As a biometric feature, eye tracking data, when compared to an iris scan or a fingerprint image, offers more security. An iris scan is usually a simple two-dimensional image, unless a specialized device for three-dimensional scanning is used, and a fingerprint is usually just another two-dimensional image with most of the additional metrics used by the fingerprint scanners available to the public being just simple tests for temperature or pulse. In general, the popular human biometrics are two-dimensional and can be easily captured and reproduced by a simple camera. Eye tracking data is still susceptible to most vulnerabilities two-dimensional biometrics offer, however, eye tracking data has to be taken over time, which introduces a third

## Chapter 5. Discussion

time dimension to this biometric feature, making it more secure due to the more complicated techniques required for forging, including the necessity of a continuous recording over time, compared to capturing just a single moment required for forging fingerprints and iris scans. The best user authentication and verification methods still consist of something a user knows or something a user has, and eye tracking data, even with the extra forging barriers, cannot provide proper authentication guarantee.

In the future, eye tracking data could potentially be processed by the device or just the browser, which would offer web applications some sort of API to access the gaze location for the improvement of user experience, or just for the user convenience, without the risk of the footage from the user web camera being sent outside the device. A great example is another biometric feature used in a low-risk application by being processed inside the device – the fingerprint. User fingerprints are nowadays common to be used while unlocking a cell phone – a low-risk field where the usage of a biometric feature seems excessive. However, it improves the user experience and is convenient for the user. Eye trackers could easily be implemented in phones and other devices, just to improve user convenience. In case of a wide usage of eye trackers, authenticating users based on their scanpath might become popular.

## 6 Conclusion

In order to research the possible improvement in browser fingerprinting using eye tracking data, an eye-tracking website has been made. The website allowed users to browse other websites while tracking their eyes. The eye tracking data, combined with the browser fingerprinting data has been saved into a database. The datasets have been imported to MatLab and converted into a numeric value in order to be easier processed by the machine learning cross-validation algorithms.

The datasets were divided into different testing datasets: one dataset with just the browser fingerprint, one dataset with only the eye tracking data and one browser with both datasets combined. Each of the datasets has been tested with a fine tree, a coarse tree and a bagged tree cross-validation algorithm. The results of the tests can be seen on Figure 6.1:



Figure 6.1: The resulting accuracy determined by cross-validation using different algorithms. The blue line represents the fine and coarse tree algorithms, the green line represents the bagged tree algorithm

With all three decision tree algorithms, the classification accuracy has been improved from 75% to 80%, and the variance in the accuracy depending on test subject has been dropped significantly.

Overall, based on the experimental data used in this study, browser fingerprinting can be improved using eye tracking data used as an input to machine learning decision tree classification algorithms



# Bibliography

- [17] Learn x in y minutes: Scenic programming language tours. [www.learnxinyminutes.com](http://www.learnxinyminutes.com). Accessed: 2020-10-21.
- [18] Financial times. [www.ft.com](http://www.ft.com). Accessed: 2020-10-21.
- [19] Schwäbisches tagblatt tübingen. [www.tagblatt.de/Nachrichten/Tuebingen](http://www.tagblatt.de/Nachrichten/Tuebingen). Accessed: 2020-10-21.
- [20] Marinetraffic: Global ship tracking intelligence. [www.marinetraffic.com](http://www.marinetraffic.com). Accessed: 2020-10-21.
- [21] Global cases of covid-19. [www.studylib.net/coronavirus](http://www.studylib.net/coronavirus). Accessed: 2020-10-21.
- [22] Awardspace.com: Free web hosting with php, mysql. [www.awardspace.com](http://www.awardspace.com). Accessed: 2020-10-21.
- [23] X-frame-options. [www.developer.mozilla.org/de/docs/Web/HTTP/Headers/X-Frame-Options](http://www.developer.mozilla.org/de/docs/Web/HTTP/Headers/X-Frame-Options).
- [24] Html iframe tag. [www.w3schools.com/tags/tag\\_iframe.ASP](http://www.w3schools.com/tags/tag_iframe.ASP). Accessed: 2020-10-21.
- [80] Html <div> tag. [www.w3schools.com/tags/tag\\_div.ASP/](http://www.w3schools.com/tags/tag_div.ASP/). Accessed: 2020-10-21.
- [82] Fingerprintjs. [www.github.com/fingerprintjs/fingerprintjs/](http://www.github.com/fingerprintjs/fingerprintjs/). Accessed: 2020-10-21.
- [83] What is the html dom?. [www.w3schools.com/whatis/whatis\\_htmlldom.asp/](http://www.w3schools.com/whatis/whatis_htmlldom.asp/). Accessed: 2020-10-21.
- [84] Html hidden attribute. [www.w3schools.com/tags/att\\_global\\_hidden.asp](http://www.w3schools.com/tags/att_global_hidden.asp). Accessed: 2020-10-21.
- [AP] Ida De Smet Xander Koo James Tompkin Jeff Huangh A. Papoutsaki, Aaron Gokaslan. Eample of using webgazer. [www.webgazer.cs.brown.edu/](http://www.webgazer.cs.brown.edu/). Accessed: 2020-10-21.
- [BFG<sup>+</sup>16] H. Bahmani, W. Fuhl, E. Gutierrez, G. Kasneci, E. Kasneci, and S. Wahl. Feature-based attentional influences on the accommodation response. In *Vision Sciences Society Annual Meeting Abstract*, 2016.

## Bibliography

- [BFGI11] Károly Boda, Ádám Máté Földes, Gábor György Gulyás, and Sándor Imre. User tracking on the web via cross-browser fingerprinting. In *Nordic conference on secure it systems*, pages 31–46. Springer, 2011.
- [BH07] Kenrick Mock Bogdan Hoanca. Methods and systems for multiple factor authentication using gaze tracking and iris scanning, US7986816B1, 2007.
- [BPS72] Monte Buchsbaum, Adolf Pfefferbaum, and Richard Stillman. Individual differences in eye-movement patterns. *Perceptual and Motor Skills*, 35(3):895–901, 1972. PMID: 4643983.
- [CAS01] Monchu Chen, John Anderson, and Myeong Sohn. What can a mouse cursor tell us more? correlation of eye/mouse movements on web browsing. *Proceedings of CHI Extended Abstracts*, pages 281–282, 01 2001.
- [CCH<sup>+</sup>17] Tim Chuk, Kate Crookes, William G. Hayward, Antoni B. Chan, and Janet H. Hsiao. Hidden markov model analysis reveals the advantage of analytic eye movement patterns in face recognition across cultures. *Cognition*, 169:102 – 117, 2017.
- [CGN<sup>+</sup>15a] Virginio Cantoni, Chiara Galdi, Michele Nappi, Marco Porta, and Daniel Riccio. Gant: Gaze analysis technique for human identification. *Pattern Recognition*, 48, 04 2015.
- [CGN<sup>+</sup>15b] Virginio Cantoni, Chiara Galdi, Michele Nappi, Marco Porta, and Daniel Riccio. Gant: Gaze analysis technique for human identification. *Pattern Recognition*, 48, 04 2015.
- [Cho20] Davuluri Hemanth Chowdary. Decision trees explained with a practical example. [www.towardsai.net/p/programming/decision-trees-explained-with-a-practical-example-fe47872d3b53/](http://www.towardsai.net/p/programming/decision-trees-explained-with-a-practical-example-fe47872d3b53/), 2020. Accessed: 2020-10-21.
- [CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September 1995.
- [DB12] Ali Darwish and Emad Bataineh. Eye tracking analysis of browser security indicators. pages 1–6, 12 2012.
- [EFK17] Shahram Eivazi, Wolfgang Fuhl, and Enkelejda Kasneci. Towards intelligent surgical microscopes: Surgeons gaze and instrument tracking. In *Proceedings of the 22st International Conference on Intelligent User Interfaces, IUI 2017*. ACM, 03 2017.
- [EHF<sup>+</sup>17] S. Eivazi, A. Hafez, W. Fuhl, H. Afkari, E. Kasneci, M. Lehecka, and R. Bednarik. Optimal eye movement strategies: a comparison of neurosurgeons gaze patterns when using a surgical microscope. *Acta Neurochirurgica*, 2017.

- [EL14] Zeno Rocha Pablo Carvalho Maira Bello Eduardo Lundgren, Thiago Rocha. tracking.js: A modern approach for computer vision on the web. trackingjs.com, 2014. Accessed: 2020-10-21.
- [EMD<sup>+</sup>18] N. Eiselt, D. Muench, A. Dochhan, H. Griesser, M. Eiselt, J. J. V. Olmos, I. T. Monroy, and J. Elbers. Performance comparison of 112-gb/s dmt, nyquist pam4, and partial-response pam4 for future 5g ethernet-based fronthaul architecture. *Journal of Lightwave Technology*, 36(10):1807–1814, 2018.
- [ESF<sup>+</sup>17] Shahram Eivazi, Michael Slupina, Wolfgang Fuhl, Hoorieh Afkari, Ahmad Hafez, and Enkelejda Kasneci. Towards automatic skill evaluation in microsurgery. In *Proceedings of the 22st International Conference on Intelligent User Interfaces, IUI 2017*. ACM, 03 2017.
- [FBH<sup>+</sup>19] Wolfgang Fuhl, Efe Bozkir, Benedikt Hosp, Nora Castner, David Geisler, Thiago C., and Enkelejda Kasneci. Encodji: Encoding gaze data into emoji space for an amusing scanpath classification approach ;). In *Eye Tracking Research and Applications*, 2019.
- [FBK20] Wolfgang Fuhl, Efe Bozkir, and Enkelejda Kasneci. Reinforcement learning for the privacy preservation and manipulation of eye tracking data. *arXiv preprint arXiv:2002.06806*, 08 2020.
- [FCK18a] W. Fuhl, N. Castner, and E. Kasneci. Histogram of oriented velocities for eye movement detection. In *International Conference on Multimodal Interaction Workshops, ICMIW*, 2018.
- [FCK18b] W. Fuhl, N. Castner, and E. Kasneci. Rule based learning for eye movement type detection. In *International Conference on Multimodal Interaction Workshops, ICMIW*, 2018.
- [FCK<sup>+</sup>19] W. Fuhl, N. Castner, T. C. Kübler, A. Lotz, W. Rosenstiel, and E. Kasneci. Ferns for area of interest free scanpath classification. In *Proceedings of the 2019 ACM Symposium on Eye Tracking Research & Applications (ETRA)*, 06 2019.
- [FCZ<sup>+</sup>18] W. Fuhl, N. Castner, L. Zhuang, M. Holzer, W. Rosenstiel, and E. Kasneci. Mam: Transfer learning for fully automatic video annotation and specialized detector creation. In *International Conference on Computer Vision Workshops, ICCVW*, 2018.
- [FEH<sup>+</sup>18] W. Fuhl, S. Eivazi, B. Hosp, A. Eivazi, W. Rosenstiel, and E. Kasneci. Bore: Boosted-oriented edge optimization for robust, real time remote pupil center detection. In *Eye Tracking Research and Applications, ETRA*, 2018.
- [FGK20a] W. Fuhl, H. Gao, and E. Kasneci. Neural networks for optical vector

## Bibliography

- and eye ball parameter estimation. In *ACM Symposium on Eye Tracking Research & Applications, ETRA 2020*. ACM, 01 2020.
- [FGK20b] W. Fuhl, H. Gao, and E. Kasneci. Tiny convolution, decision tree, and binary neuronal networks for robust and real time pupil outline estimation. In *ACM Symposium on Eye Tracking Research & Applications, ETRA 2020*. ACM, 01 2020.
- [FGRK19] W. Fuhl, D. Geisler, W. Rosenstiel, and E. Kasneci. The applicability of cycle gans for pupil and eyelid segmentation, data generation and image refinement. In *International Conference on Computer Vision Workshops, ICCVW*, 11 2019.
- [FGS<sup>+</sup>18] W. Fuhl, D. Geisler, T. Santini, T. Appel, W. Rosenstiel, and E. Kasneci. Cbf:circular binary features for robust and real-time pupil center detection. In *ACM Symposium on Eye Tracking Research & Applications*, 06 2018.
- [FK18] W. Fuhl and E. Kasneci. Eye movement velocity and gaze data generator for evaluation, robustness testing and assess of eye tracking software and visualization tools. In *Poster at Egocentric Perception, Interaction and Computing, EPIC*, 2018.
- [FK19] W. Fuhl and E. Kasneci. Learning to validate the quality of detected landmarks. In *International Conference on Machine Vision, ICMV*, 11 2019.
- [FK20a] Wolfgang Fuhl and Enkelejda Kasneci. Rotated ring, radial and depth wise separable radial convolutions. *arXiv*, 08 2020.
- [FK20b] Wolfgang Fuhl and Enkelejda Kasneci. Weight and gradient centralization in deep neural networks. *arXiv*, 08 2020.
- [FKB<sup>+</sup>18] W. Fuhl, T. C. Kübler, H. Brinkmann, R. Rosenberg, W. Rosenstiel, and E. Kasneci. Region of interest generation algorithms for eye tracking data. In *Third Workshop on Eye Tracking and Visualization (ETVIS), in conjunction with ACM ETRA*, 06 2018.
- [FKH<sup>+</sup>17] W. Fuhl, T. C. Kübler, D. Hospach, O. Bringmann, W. Rosenstiel, and E. Kasneci. Ways of improving the precision of eye tracking data: Controlling the influence of dirt and dust on pupil detection. *Journal of Eye Movement Research*, 10(3), 05 2017.
- [FKRK20] W. Fuhl, G. Kasneci, W. Rosenstiel, and E. Kasneci. Training decision trees as replacement for convolution layers. In *Conference on Artificial Intelligence, AAI*, 02 2020.
- [FKS<sup>+</sup>15a] W. Fuhl, T. C. Kübler, K. Sippel, W. Rosenstiel, and E. Kasneci. Arbitrarily shaped areas of interest based on gaze density gradient. In *European Conference on Eye Movements, ECEM 2015*, 08 2015.

- [FKS<sup>+</sup>15b] W. Fuhl, T. C. Kübler, K. Sippel, W. Rosenstiel, and E. Kasneci. Excuse: Robust pupil detection in real-world scenarios. In *16th International Conference on Computer Analysis of Images and Patterns (CAIP 2015)*, 09 2015.
- [FKSK18] W. Fuhl, T. Kübler, T. Santini, and E. Kasneci. Automatic generation of saliency-based areas of interest. In *Symposium on Vision, Modeling and Visualization (VMV)*, 09 2018.
- [FMSM09] Clinton Fookes, Anthony Maeder, Sridha Sridharan, and George Mamic. Gaze based personal identification. *Behavioral Biometrics for Human Identification: Intelligent Applications*, 01 2009.
- [FRE20] Wolfgang Fuhl, Yao Rong, and Kasneci Enkelejda. Fully convolutional neural networks for raw eye tracking data segmentation, generation, and reconstruction. In *Proceedings of the International Conference on Pattern Recognition*, pages 0–0, 2020.
- [FRK19] W. Fuhl, W. Rosenstiel, and E. Kasneci. 500,000 images closer to eyelid and pupil segmentation. In *Computer Analysis of Images and Patterns, CAIP*, 11 2019.
- [FRM<sup>+</sup>20] Wolfgang Fuhl, Yao Rong, Thomas Motz, Michael Scheidt, Andreas Hartel, Andreas Koch, and Enkelejda Kasneci. Explainable online validation of machine learning models for practical applications. *arXiv*, 08 2020.
- [FSG<sup>+</sup>16] W. Fuhl, T. Santini, D. Geisler, T. C. Kübler, W. Rosenstiel, and E. Kasneci. Eyes wide open? eyelid location and eye aperture estimation for pervasive eye tracking in real-world scenarios. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct publication – PETMEI 2016*, 09 2016.
- [FSG<sup>+</sup>17] W. Fuhl, T. Santini, D. Geisler, T. C. Kübler, and E. Kasneci. Eyelad: Remote eye tracking image labeling tool. In *12th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017)*, 02 2017.
- [FSK17a] W. Fuhl, T. Santini, and E. Kasneci. Fast and robust eyelid outline and aperture detection in real-world scenarios. In *IEEE Winter Conference on Applications of Computer Vision (WACV 2017)*, 03 2017.
- [FSK17b] W. Fuhl, T. Santini, and E. Kasneci. Fast camera focus estimation for gaze-based focus control. In *CoRR*, 2017.
- [FSK<sup>+</sup>18] W. Fuhl, T. Santini, T. Kuebler, N. Castner, W. Rosenstiel, and E. Kasneci. Eye movement simulation and detector creation to reduce laborious parameter adjustments. *arXiv preprint arXiv:1804.00970*, 2018.

## Bibliography

- [FSR<sup>+</sup>16] W. Fuhl, T. Santini, C. Reichert, D. Claus, A. Herkommer, H. Bahmani, K. Rifai, S. Wahl, and E. Kasneci. Non-intrusive practitioner pupil detection for unmodified microscope oculars. *Elsevier Computers in Biology and Medicine*, 79:36–44, 12 2016.
- [FTBK16] Wolfgang Fuhl, Marc Tonsen, Andreas Bulling, and Enkelejda Kasneci. Pupil detection for head-mounted eye tracking in the wild: An evaluation of the state of the art. In *Machine Vision and Applications*, pages 1–14, 06 2016.
- [Fuh19] W. Fuhl. *Image-based extraction of eye features for robust eye tracking*. PhD thesis, University of Tübingen, 04 2019.
- [Fuh20] Wolfgang Fuhl. From perception to action using observed actions to learn gestures. *User Modeling and User-Adapted Interaction*, pages 1–18, 08 2020.
- [GA10] Qi Guo and Eugene Agichtein. Towards predicting web searcher gaze position from mouse movements. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems, CHI EA '10*, page 3601–3606, New York, NY, USA, 2010. Association for Computing Machinery.
- [GFSK17] D. Geisler, W. Fuhl, T. Santini, and E. Kasneci. Saliency sandbox: Bottom-up saliency framework. In *12th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017)*, 02 2017.
- [GH10] Joseph Goldberg and Jonathan Helfman. Visual scanpath representation. pages 203–210, 01 2010.
- [GHJP00] Steven Glass, Tom Hiller, Stuart Jacobs, and C Perkins. Mobile ip authentication, authorization, and accounting requirements. *Work in Progress*, 2000.
- [Gun20] Satish Gunjal. Support vector machines. [www.satishgunjal.com/svm//](http://www.satishgunjal.com/svm//), 2020. Accessed: 2020-10-21.
- [Gup17] Prashant Gupta. Decision trees in machine learning. [www.towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052/](http://www.towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052/), 2017. Accessed: 2020-10-21.
- [Her15] Lucas A. Herrera. Authentication method using multi-factor eye gaze, US20150302252A1, 2015.
- [HH16] Taylor Hayes and John Henderson. Eye movement patterns during scene viewing predict individual differences. *Journal of Vision*, 16:329, 09 2016.
- [HK12] Arthur Hoerl and Robert Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67, 04 2012.

- [HLMB18a] Sabrina Hoppe, Tobias Loetscher, Stephanie A. Morey, and Andreas Bulling. Eye movements during everyday behavior predict personality traits. *Frontiers in Human Neuroscience*, 12:105, 2018.
- [HLMB18b] Sabrina Hoppe, Tobias Loetscher, Stephanie A. Morey, and Andreas Bulling. Eye movements during everyday behavior predict personality traits. *Frontiers in Human Neuroscience*, 12:105, 2018.
- [HPW11a] David Hauger, Alexandros Paramythis, and Stephan Weibelzahl. Using browser interaction data to determine page reading behavior. In Joseph A. Konstan, Ricardo Conejo, José L. Marzo, and Nuria Oliver, editors, *User Modeling, Adaption and Personalization*, pages 147–158, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
- [HPW11b] David Hauger, Alexandros Paramythis, and Stephan Weibelzahl. Using browser interaction data to determine page reading behavior. In Joseph A. Konstan, Ricardo Conejo, José L. Marzo, and Nuria Oliver, editors, *User Modeling, Adaption and Personalization*, pages 147–158, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.
- [HWB12] Jeff Huang, Ryen White, and Georg Buscher. User see, user point: Gaze and cursor alignment in web search. *Conference on Human Factors in Computing Systems - Proceedings*, 05 2012.
- [JCG17] Cem Keskin John C. Gordon. Gaze-based authentication, US20180018514A1, 2017.
- [Kas04a] Pawel Kasprowski. *Human identification using eye movements*. PhD thesis, 11 2004.
- [Kas04b] Pawel Kasprowski. *Human identification using eye movements*. PhD thesis, 11 2004.
- [KO04] Pawel Kasprowski and Józef Ober. Eye movements in biometrics. In Davide Maltoni and Anil K. Jain, editors, *Biometric Authentication*, pages 248–258, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
- [KSF<sup>+</sup>15] T. C. Kübler, K. Sippel, W. Fuhl, G. Schievelbein, J. Aufreiter, R. Rosenberg, W. Rosenstiel, and E. Kasneci. *Analysis of eye movements with Eyetrace*, volume 574. Biomedical Engineering Systems and Technologies. Communications in Computer and Information Science (CCIS). Springer International Publishing, 2015.
- [Kul17] Mayur Kulkarni. Decision trees for classification: A machine learning algorithm. [www.xoriant.com/blog/product-engineering/decision-trees-machine-learning-algorithm.html/](http://www.xoriant.com/blog/product-engineering/decision-trees-machine-learning-algorithm.html/), 2017. Accessed: 2020-10-21.
- [KV11] Victor Kuperman and Julie A. Van Dyke. Effects of individual differences

## Bibliography

- in verbal skills on eye-movement patterns during sentence reading. *Journal of Memory and Language*, 65(1):42 – 73, 2011.
- [LD14] Daniel Liebling and Susan Dumais. Gaze and mouse coordination in everyday work. *UbiComp 2014 - Adjunct Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 1141–1150, 09 2014.
- [LVG08] Beatriz Luna, Katerina Velanova, and Charles Geier. Development of eye-movement control. *Brain and cognition*, 68:293–308, 11 2008.
- [LW01] Gerald Lohse and D. Wu. Eye movement patterns on chinese yellow pages advertising. *Electronic Markets*, 11:87–96, 04 2001.
- [MFS04] Anthony Maeder, C. Fookes, and Sridharan Subramanian. Gaze based user authentication for personal computer applications. pages 727 – 730, 11 2004.
- [MKMS17] Raphael Menges, Chandan Kumar, Daniel Müller, and Korok Sengupta. Gazetheweb: A gaze-controlled web browser. pages 1–2, 04 2017.
- [MS16] Henrik Jönsson Anders Vennström Erland George-Svahn John Elvesjö Mårten Skogö, Richard HAINZL. Identification and/or authentication of a user using gaze information, US10192109B2, 2016.
- [NL15] Efi Nisiforou and Andrew Laghos. Field dependence–independence and eye movement patterns: Investigating users’ differences through an eye tracking study. *Interacting with Computers*, 28, 06 2015.
- [OLSM07] Erik Olsen, Suzanne Lee, and Bruce Simons-Morton. Eye movement patterns for novice teen drivers does 6 months of driving experience make a difference? *Transportation research record*, 2009:8–14, 12 2007.
- [OS07] Haruhisa Okawa and Alapakkam P. Sampath. Optimization of single-photon response transmission at the rod-to-rod bipolar synapse. *Physiology*, 22(4):279–286, 2007. PMID: 17699881.
- [PD01] Fitzpatrick D Purves D, Augustine GJ. Neuroscience. 2nd edition. sunderland (ma): Sinauer associates; 2001. neural control of saccadic eye movements. <https://www.ncbi.nlm.nih.gov/books/NBK10992/>, 2001. Accessed: 2020-10-21.
- [Per] Andrea Perlato. Introduction to support vector machine.
- [PJD10] Andrea L. Patalano, Barbara J. Juhasz, and Joanna Dicke. The relationship between indecisiveness and eye movement patterns in a decision making informational search task. *Journal of Behavioral Decision Making*, 23(4):353–368, 2010.

- [PSL<sup>+</sup>16a] Alexandra Papoutsaki, Patsorn Sangkloy, James Laskey, Nediya Daskalova, Jeff Huang, and James Hays. Webgazer: Scalable webcam eye tracking using user interactions. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence-IJCAI 2016*, 2016.
- [PSL<sup>+</sup>16b] Alexandra Papoutsaki, Patsorn Sangkloy, James Laskey, Nediya Daskalova, Jeff Huang, and James Hays. Webgazer: Scalable webcam eye tracking using user interactions. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3839–3845. AAAI, 2016.
- [RFAS08] Kerry Rodden, Xin Fu, Anne Aula, and Ian Spiro. Eye-mouse coordination patterns on web search pages. pages 2997–3002, 01 2008.
- [RFZ01] Dragomir R Radev, Weiguo Fan, and Zhu Zhang. Webinessence: A personalized web-based multi-document summarization and recommendation system. In *NAACL Workshop on Automatic Summarization*. Citeseer, 2001.
- [RPC01] Robert Reeder, Peter Pirolli, and Stuart Card. Webeyemapper and weblogger: Tools for analyzing eye tracking data collected in web-use studies. *Conference on Human Factors in Computing Systems - Proceedings*, 04 2001.
- [RPTH17] Vijay Rajanna, Seth Polsley, Paul Taele, and Tracy Hammond. A gaze gesture-based user authentication system to counter shoulder-surfing attacks. pages 1978–1986, 05 2017.
- [Sch11] Daniel Schacter. *Psychology*. Worth Publishers, New York, NY, 2011.
- [SCM<sup>+</sup>09] Ashkan Soltani, Shannon Canty, Quentin Mayo, Lauren Thomas, and Chris Jay Hoofnagle. Flash cookies and privacy. *Available at SSRN 1446862*, 2009.
- [Sta06] Cecie Starr. *Biology : concepts and applications*. Thomson, Brooks/Cole, Belmont, CA, 2006.
- [STD19] Bahman Sargezeh, Niloofar Tavakoli, and mohammad reza Daliri. Gender-based eye movement differences in passive indoor picture viewing: An eye-tracking study. *Physiology and Behavior*, 206, 03 2019.
- [TC92] P. . Tu and J. . Chung. A new decision-tree classification algorithm for machine learning. In *Proceedings Fourth International Conference on Tools with Artificial Intelligence TAI '92*, pages 370–377, 1992.
- [TGS<sup>+</sup>14] Hien Thi Thu Truong, Xiang Gao, Babins Shrestha, Nitesh Saxena, N Asokan, and Petteri Nurmi. Comparing and fusing different sensor modalities for relay attack resistance in zero-interaction authentica-

## Bibliography

- tion. In *2014 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 163–171. IEEE, 2014.
- [Tsc12] Martin Tschirsich. Js-objectdetect: Computer vision in your browser - javascript realtime object de- tection. <https://github.com/mtschirs/js-objectdetect>, 2012. Accessed: 2020-10-21.
- [Wik20a] Wikipedia. Abducens nerve Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Abducens%20nerve&oldid=937216453>, 2020. [Online; accessed 22-October-2020].
- [Wik20b] Wikipedia. Ampel Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Ampel&oldid=893133797>, 2020. [Online; accessed 22-October-2020].
- [Wik20c] Wikipedia. Aqueous humour Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Aqueous%20humour&oldid=971978260>, 2020. [Online; accessed 22-October-2020].
- [Wik20d] Wikipedia. Augenmuskeln Wikipedia, the free encyclopedia. <http://de.wikipedia.org/w/index.php?title=Augenmuskeln&oldid=204576927>, 2020. [Online; accessed 22-October-2020].
- [Wik20e] Wikipedia. Bootstrap aggregating Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Bootstrap%20aggregating&oldid=979505674>, 2020. [Online; accessed 22-October-2020].
- [Wik20f] Wikipedia. Color vision Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Color%20vision&oldid=982780620>, 2020. [Online; accessed 22-October-2020].
- [Wik20g] Wikipedia. Cone cell Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Cone%20cell&oldid=982907805>, 2020. [Online; accessed 22-October-2020].
- [Wik20h] Wikipedia. Cranial nerves Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Cranial%20nerves&oldid=984276142>, 2020. [Online; accessed 22-October-2020].
- [Wik20i] Wikipedia. Decision tree learning Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Decision%20tree%20learning&oldid=983310740>, 2020. [Online; accessed 22-October-2020].
- [Wik20j] Wikipedia. Decision tree Wikipedia, the free encyclope-

- dia. <http://en.wikipedia.org/w/index.php?title=Decision%20tree&oldid=983253586>, 2020. [Online; accessed 22-October-2020].
- [Wik20k] Wikipedia. Ensemble learning Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Ensemble%20learning&oldid=977960246>, 2020. [Online; accessed 22-October-2020].
- [Wik20l] Wikipedia. Extraocular muscles Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Extraocular%20muscles&oldid=984293896>, 2020. [Online; accessed 22-October-2020].
- [Wik20m] Wikipedia. Eye movement Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Eye%20movement&oldid=984295653>, 2020. [Online; accessed 22-October-2020].
- [Wik20n] Wikipedia. Fovea centralis Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Fovea%20centralis&oldid=983751935>, 2020. [Online; accessed 22-October-2020].
- [Wik20o] Wikipedia. Human eye Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Human%20eye&oldid=984322220>, 2020. [Online; accessed 22-October-2020].
- [Wik20p] Wikipedia. Iris (anatomy) Wikipedia, the free encyclopedia. [http://en.wikipedia.org/w/index.php?title=Iris%20\(anatomy\)&oldid=982885955](http://en.wikipedia.org/w/index.php?title=Iris%20(anatomy)&oldid=982885955), 2020. [Online; accessed 22-October-2020].
- [Wik20q] Wikipedia. Light Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Light&oldid=970084883>, 2020. [Online; accessed 22-October-2020].
- [Wik20r] Wikipedia. Medial rectus muscle Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Medial%20rectus%20muscle&oldid=907653149>, 2020. [Online; accessed 22-October-2020].
- [Wik20s] Wikipedia. Oculomotor nerve Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Oculomotor%20nerve&oldid=984340318>, 2020. [Online; accessed 22-October-2020].
- [Wik20t] Wikipedia. Ophthalmic artery Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Ophthalmic%20artery&oldid=937990205>, 2020. [Online; accessed 22-October-2020].

## Bibliography

- [Wik20u] Wikipedia. Photoreceptor cell Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Photoreceptor%20cell&oldid=983662575>, 2020. [Online; accessed 22-October-2020].
- [Wik20v] Wikipedia. Retina Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Retina&oldid=982230533>, 2020. [Online; accessed 22-October-2020].
- [Wik20w] Wikipedia. Rod cell Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Rod%20cell&oldid=982708791>, 2020. [Online; accessed 22-October-2020].
- [Wik20x] Wikipedia. Sehnerv Wikipedia, the free encyclopedia. <http://de.wikipedia.org/w/index.php?title=Sehnerv&oldid=204072754>, 2020. [Online; accessed 22-October-2020].
- [Wik20y] Wikipedia. Superior rectus muscle Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Superior%20rectus%20muscle&oldid=870903714>, 2020. [Online; accessed 22-October-2020].
- [Wik20z] Wikipedia. Support vector machine Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Support%20vector%20machine&oldid=984421646>, 2020. [Online; accessed 22-October-2020].
- [WMH11] Justin Weaver, Kenrick Mock, and Bogdan Hoanca. Gaze-based password authentication through automatic clustering of gaze points. pages 2749–2754, 10 2011.
- [ZC16] Yu-Qian Zhu and Jung-Hua Chang. The key role of relevance in personalized advertisement: Examining its impact on perceptions of privacy invasion, self-awareness, and continuous use intentions. *Computers in Human Behavior*, 65:442–447, 2016.
- [ZJ12] Xiaoling Zheng and Jidong Jin. Research for the application and safety of md5 algorithm in password authentication. In *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery*, pages 2216–2219. IEEE, 2012.

# List of Figures

2.1	An example of a simple decision tree, tasked with making a decision based on weather [Kul17]	11
2.2	The training dataset used for the decision tree displayed on Figure 2.1 [Kul17]	12
2.3	A manually made decision tree [Wik20j]	13
2.4	A margin classifier [Gun20]	14
2.5	A margin classifier being affected by an outlier (the red circle close to the green circles) causes the new black sample to be classified as red despite being closer to the green group [Gun20]	14
2.6	The support vector [Gun20]	14
2.7	The new black sample will be classified as green because of the support vector being calculated to be between the majority of the elements of each class [Gun20]	14
2.8	Two groups of samples are distributed in a fashion not allowing for a hyperplane to divide them in a one-dimensional space [Per]	15
2.9	The samples from Figure 2.8 have been elevated into a higher dimension, with a new sample being correctly classified, and a hyperplane dividing the two groups [Per]	15
2.10	The visible spectrum of light, with the wavelength in nm [Wik20q]	16
2.11	The whole spectrum of electromagnetic waves [Wik20q]	16
2.12	Traffic lights provide critical information to humans operating motor vehicles via the visible part of the electromagnetic spectrum [Wik20b]	17
2.13	The general structure of a human eye [Wik20o]	18
2.14	Different layers of the 0.5mm thick retina [Wik20v]	19
2.15	Sensitivity intensity of different cones to different wavelengths [Wik20u]	20
2.16	Location of the eye inside it's orbits, with muscles and a layer of fat as well as the optical nerve being visible [Wik20o]	21
2.17	Eye muscles allow the eye to be rotated in any direction [Wik20o]	21
2.18	Eye from above, the superior rectus in the foreground and inferior rectus in the background, the pair of antagonist muscles responsible for the ability of an eye to look upwards and downwards [Wik20y]	23
2.19	Nerves around the eye [Wik20x]	24

## List of Figures

2.20	The neuron bursting rate is depicted as the frequency of black lines corresponding to eye movement. The baseline frequency can be seen with the eye holding position (horizontal line in the blue area), while the movement distance is clearly correlated to the length of a bursting event. [PD01]	24
2.21	An example of WebGazer tracking the gaze on a website. [AP]	26
2.22	An example of a user scanpath in a web browser [GH10]	28
3.1	An example of fixations on a face stimuli with grid [CGN <sup>+</sup> 15b].	29
3.2	Constant stimuli (b) used by [MFS04], where the similarity of a scanpath (a) is used as input data.	30
3.3	An example of random stimuli used by [Kas04b], the number under each stimuli is the time needed for the user experimented on to move the gaze to the position on a screen.	31
3.4	An example of an iframe being used. FT.com is being displayed inside an iframe element on www.w3schools.com	32
3.5	An example of an x-frame-origin not allowing usage of the website inside an iframe	33
3.6	Calibration stage, the estimated gaze is shown as the red dot	33
3.7	The request to use the webcam handled by the browser and the JS-alert window asking for a name while displaying the unique ID	34
3.8	The user ID is the only key for the fixations and the fingerprints in this database. Without the key, it is impossible to get the correct fixations.	35
3.9	The database with the user names and their browser fingerprints. Users rarely bother to use an actual name, which is why an ID is used everywhere.	37
3.10	The small round object with a number on it is used for calibration	37
3.11	The debugging tool used to visualize fixations for a certain ID from the database, show.html. The fixations for ID 693 are shown.	38
3.12	The gaze recording stage, with five websites available with five orange buttons buttons in the corners	40
4.1	The confusion matrix and the accuracy of the Fine Decision Tree algorithm for browser data	43
4.2	The confusion matrix and the accuracy of the Fine Decision Tree algorithm for eye tracking data	44
4.3	The confusion matrix and the accuracy of the Fine Decision Tree algorithm for combined eye tracking and browser data	45
4.4	The confusion matrix and accuracy for a Coarse Decision Tree algorithm used on pure browser data	46
4.5	The confusion matrix and accuracy for a Coarse Decision Tree algorithm used on pure eye tracking data	47
4.6	The confusion matrix and accuracy for a Coarse Decision Tree algorithm used on both eye tracking and browser data	48

4.7	The confusion matrix and accuracy for a Bagged Tree algorithm used with browser data . . . . .	49
4.8	The confusion matrix and accuracy for a Bagged Tree algorithm for pure eye tracking data . . . . .	50
4.9	The confusion matrix and accuracy for a Bagged Tree algorithm for combined data . . . . .	51
4.10	Classification accuracy based on the data types across different algorithms. . . . .	52
5.1	A high-resolution image of a human iris with the human eye around the iris [Wik20p]. This biometric feature cannot be captured by modern technology available in every laptop, opposed to the eye movements	54
6.1	The resulting accuracy determined by cross-validation using different algorithms. The blue line represents the fine and coarse tree algorithms, the green line represents the bagged tree algorithm . . . . .	57