EyeGaze: Automated sheet music page-turning using gaze detection

by

Christopher Nixon

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master in Computer Science (MCS)

Supervisor: Dr. Kenneth Dawson-Howe

May 2022
I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

___________________________________
Christopher Nixon
April 19th 2022
Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

______________________________
Christopher Nixon
April 19th 2022
I would firstly like to thank my supervisor Dr. Kenneth Dawson-Howe for his continued guidance, support and invaluable suggestions throughout this project.

I would also like to thank my parents David and Rhoda Nixon for their steadfast support of my education over many, many years.

Finally, a special thanks to Orlaith Dunne, Samuel Nixon and my friends for their emotional support and encouragement.

Christopher Nixon

University of Dublin, Trinity College
April 2022
EyeGaze: Automated sheet music page-turning using gaze detection

Christopher Nixon, Master in Computer Science (MCS)
University of Dublin, Trinity College, 2022

Supervisor: Dr. Kenneth Dawson-Howe

Turning pages of sheet music is a common source of irritation for musicians, and this paper aims to address this problem by developing the EyeGaze application. The EyeGaze application has been developed for a tablet device and explores the viability of using gaze detection as a method of triggering automatic page turns in a sheet music application. Creating a passive interface where page turns are triggered without conscious input from the musician will free musicians from the distraction of thinking about page turns. In an effort to understand the effect that various gaze detection solutions and page-turning systems have on the performance of the application two different gaze detection solutions and three different page-turning systems have been implemented. These systems are evaluated by a user study where both qualitative and quantitative results were obtained. The results clearly indicated that certain page-turning systems are better suited to this use case than others, and highlighted the challenges with performing gaze detection in a general environment with uncontrolled lighting or positioning of the user.
Contents

1 Introduction 2
   1.1 Background 2
   1.2 Motivation 3
   1.3 Overview 4

2 Background to page-turning solutions and gaze-detection 5
   2.1 Page-turning solutions background 5
   2.2 Gaze detection background 6

3 Design and Development of the EyeGaze app 10
   3.1 EyeGaze app architecture and design 10
      3.1.1 Gaze detection systems 12
      3.1.2 Page-turning system designs 13
   3.2 Implementation of the EyeGaze application 15
      3.2.1 Calibration 15
      3.2.2 Face and Eye segmentation 16
      3.2.3 iTracker CNN 18
      3.2.4 Implementation of scrolling page-turning system 19
      3.2.5 Implementation of single animation page-turning system 19
      3.2.6 Implementation of double animation page-turning system 20

4 Evaluation of the EyeGaze application 21
   4.1 Evaluation approach 21
      4.1.1 User study 22
      4.1.2 Metrics 23
   4.2 Results 24
   4.3 Analysis of results 28
      4.3.1 Failure cases in page-turning systems 29
      4.3.2 Gaze detection systems 31
      4.3.3 Summary 32
5 Conclusion and Future Works 33
  5.1 Conclusion 33
  5.2 Future works 33

Bibliography 35
1 Introduction

1.1 Background

Turning pages of sheet music can be a source of irritation for musicians, particularly in a practice setting. A musician will often have to lift a hand from their instrument to turn the page, in fact this is so common that sheet music publishers will sometimes arrange music so that page turns land at a more convenient time. The widespread digitisation of sheet music has offered unique opportunities in the creation of an automatic sheet music page-turning device. Tablet devices are being used to display sheet music, offering exciting new possibilities for page-turning as well as unique challenges. Physical sheet music allows the musician to have two pages of music being visible at any one time; however most tablets have a screen which is smaller in size than an A4 page. A tablet device could be placed horizontally which allows two pages to be displayed in a similar fashion to physical sheet music, however this is not a popular design decision as the reduced size of the music can render it more difficult to read. Another possibility is displaying a single sheet of music on the tablet screen at a time, creating twice as many page turns. The increasing computational power of tablets, as well as the popularity of built-in microphones and front-facing cameras allows for the reconsideration of how to handle page turns. Animations can be used to imitate the appearance of a physical page being turned, music can be scrolling horizontally or vertically, or custom animations could be used to partially turn pages. Whatever design decision is taken in how page turns are handled, to create an automatic page turner there must be a system which keeps track of the musicians place in the sheet music and appropriately triggers page turns. This dissertation will explore using a tablet's front-facing camera to perform gaze detection as a system to trigger page turns.

1.2 Motivation

As was previously mentioned, turning pages of sheet music can be a challenging and disruptive activity for musicians. This is because in order to turn to the next page of sheet music a musician must lift a hand off their instrument, potentially disrupting the
flow of the music. This is not as much of an issue during performances, where a musician is more commonly performing from memory or may have another person ready to turn the pages for them. The act of turning pages can also be a distraction for musicians. There have been several different semi-automatic page-turning devices designed for both physical and digital sheet music, but what these systems all have in common is that they must be manually triggered. One of the key focuses for this project was to design a passive interface which involved minimal active interactions from the user, thus minimising the distraction caused by page turns. There are various possibilities for such a passive page-turning interface; both in the mechanism for displaying new music on-screen as well as the system to trigger such page turns. Gaze detection was chosen as the system to trigger page turns because understanding where a user is looking on-screen allows for a passive interface to be designed, removing the distraction of page-turning from the musician. Although there has already been research focused on using gaze detection to turn the pages of sheet music (Tabone et al., 2020), the use of gaze detection performed with commercial hardware in a general setting has not been studied for this use case. As such, this dissertation aims to explore the viability of gaze detection as a trigger mechanism. Various mechanisms for displaying new music, hereafter referred to as “page-turning systems”, will also be explored to identify user preference as well as whether some are more suitable for use in conjunction with gaze detection.

1.3 Overview

This dissertation will focus on evaluating gaze detection as a trigger mechanism for sheet music page-turning, as well as comparing between different page-turning systems. An application will be developed for the iOS platform which will display sheet music to users. Two separate gaze detection systems will be implemented in the application, allowing for comparisons between the two systems. Additionally, three separate page-turning systems will be implemented. The paper is structured as follows: relevant works and knowledge on gaze detection and page-turning solutions are included in Chapter 2. Chapter 3 includes details on the design and development of the application including sections describing each of the page-turning systems. To evaluate the performance of the application in a general setting a user study was undertaken, and the
results of this user study are included in Chapter 4. An analysis of these results and a
discussion on the implications of these results are also contained in Chapter 4. In the last
chapter of this paper conclusions are drawn and potential future work in this area is
outlined.
2 Background to page-turning solutions and gaze-detection

2.1 Page-turning solutions background

Page-turning devices fall into one of two categories; mechanical or digital. Mechanical page turners operate on physical pages, automating the physical process of turning to the next page. There are many possibilities for triggering mechanical page turners in a hands-free manner such as foot/hand pedals, voice-activated systems and breath-controlled switches (Wolberg & Schipper, 2012). The other category of page turners operate on digitised sheet music typically displayed using a tablet device. These digital page turners can also be triggered by various external pedals, and voice activation systems (PageFlip, 2018) (Williams-King, 2019). Having a front-facing camera and significant processing power allows for additional page-turning triggers such as audio-based triggers, facial gestures and gaze detection (Noli Software, 2021) (phonicscore, 2021). The action that these page turners perform in order to display the next page of music differs depending on how the music is being displayed on the tablet. Typically a digital page turn is associated with an animation which brings the next page into view. This animation is quite important as it gives the user an indication that the page is turning, helping users keep track of where they are in the music. A study from 2007 found that animated page turns were preferable to static page turns, in which the page is instantly changed without an animation (Blinov, 2007).

Research in the area of using real-time audio of a performance as a trigger for page-turning focused on aligning the audio of a live performance with that of a synthesised audio file of the piece the musician is playing (Arzt et al., 2008). By aligning the two audio files, an estimate of the musician’s progress through the piece could be calculated, which in turn was used to trigger page turns.

A recent study investigated gaze-detection as a trigger for page turns, using an infrared-based eye tracking device: the SMI RED500 (Tabone et al., 2020). Kalman filtering was used to smooth noisy eye tracking measurements caused by the subject looking at their hands or instrument while using the system. High levels of successful page turns
were recorded, demonstrating that highly accurate gaze detection in combination with a modified score could produce promising results.

2.2 Gaze detection background

Gaze detection has been an area of research for many years, with one of the first large user-based studies being carried out in the late 1940s (Fitts et al., 1950). Broadly speaking the goal of gaze detection or eye tracking is to understand where a subject is looking in 3D space. There are a number of different approaches to this problem, some of which are more intrusive than others. One of the less commonly used methods is electro-oculography tracking, which consists of placing electrodes on the skin around a subject's eye (Belkhiria & Peysakhovich, 2020). These electrodes are used to measure small changes in the electric potential of the skin around a subject's eye, which are caused by the changing electrostatic field of the eye as it rotates. This is a highly intrusive technique and as such is not suited for commercialisation, although one advantage is that this technique doesn’t require a clear view of the subject's eye.

![Figure 2.1: Top Left: Infrared-based eye tracker: Tobii Pro Spectrum Eye tracker (Eizo, 2021). Bottom Left: Modern head-mounted eye tracking system developed by Eye Square (eye square, 2021). Right: Early head-mounted eye tracking system from 1967 (Encyclopædia Britannica, 2021).]

The remaining eye tracking techniques are video based, and involve analysing images of the subject’s eyes (Da Silva et al., 2007). The most intrusive of these techniques is head-mounted tracking. Head-mounted techniques involve an apparatus which includes one or more cameras and typically some sort of infrared light source. The infrared
light is shone into the subject’s eye and the reflections of this light are captured by the camera. These images are then analysed and the direction of the eye can be calculated based on the reflection of the infrared light. Head-mounted systems were originally large and cumbersome, but have been miniaturised and stylised with time. Some examples of eye tracking systems can be seen in Figure 2.1. Not all head-mounted systems use infrared light: ambient light is also used, as is the case with the EyeSquare eye tracking system depicted in Figure 2.1. These kinds of head-mounted systems analyse images of the eye and the calculated pupil direction in combination with the orientation of the camera are used to determine the gaze direction.

The remaining eye tracking systems are less intrusive but still rely on video analysis. These remaining systems can be broken down into three main categories:

- Infrared-based systems
- Geometric-based systems
- Appearance-based systems

There are non-head-mounted eye tracking systems that still use infrared light, and these are fundamentally based on the same principles of reflection from the eye. An example of one of these systems can be seen in Figure 2.1.

Geometric-based systems build a 3D model of the eye and use geometric reasoning to estimate a subject’s point of regard (Chennamma & Yuan, 2013). Facial detection is used to identify a subject’s face within an image and then features such as the eye contour and pupil location are detected. These features are used to build a 3D model of a subject’s eyes. The subject’s head pose must also be identified, as the combination of the head pose and pupil directions are used to calculate a gaze vector. These techniques use depth sensors as well as RGB video: a study on the use of gaze detection as a biometric identification of an individual combined depth-sensor information with pupil detection to perform gaze detection (Cazzato et al., 2015). The roll, pitch and yaw of the head was calculated using depth sensor information, and the intersection of the corresponding head vector produced with the plane of the depth sensor gave a rough gaze estimate. This estimate was then refined by identification of the subject’s pupils.
Appearance-based systems use ambient light and standard video of a subject. Facial-detection techniques are used to isolate a face crop from an image of a subject, then features are extracted from this face image. Examples of these features are pupil location, iris location, iris size, distance between left and right pupils and eye contours. A set of training data is used to associate these features, or indeed a crop of an eye within an image with known points of regard. This allows for the creation of a mapping from features to points of regard.

There are various approaches to solving this mapping function, such as Gaussian process regression (Sugano et al., 2012), linear regression (Lu et al., 2012), (Sugano et al., 2014) or neural networks (Demjen et al., 2011). Many earlier papers focused on person-specific models with person-dependent training datasets, but in 2013 Funes Mora et al. presented a person-independent approach to gaze estimation using a remote RGB-Depth camera (Funes Mora & Odobez, 2014). Deep-learning has been used in recent years to produce end-to-end approaches (Cazzato et al., 2020).

An influential paper written in 2015 proposed a deep-learning approach, where crops of a subject’s eyes were used as input for a convolutional neural network (CNN) (Zhang et al., 2015). This paper used a dataset consisting of 213,659 images collected from 15 different subjects under various lighting conditions.

A later paper introduced iTracker, a CNN trained on a large-scale dataset named GazeCapture (Khosla, 2016). The GazeCapture dataset contains 2,445,504 images with associated points of regard, collected from 1474 subjects. The iTracker CNN takes as input a crop of each eye, an image of the subject’s face and a binary mask that indicates the location of the subject’s face in the original image. iTracker doesn’t require person-specific calibration, although it was shown in the 2016 paper that such calibration could improve the accuracy of the gaze estimations. iTracker is one of the gaze detection systems that will be implemented in the sheet music app developed in this paper.

Something that both geometric and appearance-based approaches to gaze estimation have in common is that they require segmentation of faces and eyes from images. There are several approaches to facial detection but one that is particularly relevant to this paper is the approach implemented in the built-in Vision libraries offered by Apple on their iOS platform. Apple uses a deep convolutional network for face detection.
which is optimised for on-device usage, minimising the memory footprint and GPU usage of the network (Apple, 2017). Optimising performance of face and eye segmentation from images is essential if gaze detection is to be performed in real time.
3 Design and Development of the EyeGaze app

3.1 EyeGaze app architecture and design

There were a number of important design decisions to be considered before any development of the EyeGaze application, such as the target operating system and feature set. It was decided to develop EyeGaze for the iOS platform. One of the reasons for this decision is the previously mentioned Vision framework which would simplify the task of performing gaze detection in real time. Another important consideration during the design stage was which file formats would be supported. Music that is being digitally displayed is typically stored in either the MusicXML or PDF format, and for the EyeGaze app only PDF files are supported. Although the MusicXML format provides more scope for incorporating audio as an input for understanding a user’s place in the music, this was not the focus of this paper. Additionally it was considered that sheet music is more commonly stored in a PDF format, and as such PDF files are displayed within the app using the PDFKit library. To avoid complications with importing or scanning PDF’s into the app, PDF files that were available to the user would be bundled with the app at compile time.

Figure 3.1: System architecture diagram (Khosla, 2016).
In order to compare between gaze detection systems two separate systems would be incorporated into the app, with the user having the ability to switch between the two systems. The user is also able to switch between three different page-turning methods that were implemented, allowing for comparisons between the methods.

With these requirements in mind, the overall architecture can be seen in Figure 3.1. The two key actions that a user could take are viewing a PDF or calibrating the gaze detection systems. Including a calibration flow is not strictly necessary for the basic functionality of the gaze-detection systems as both of the gaze detection systems provide gaze estimations without user-specific calibration, but including calibration allows for several advantages. The calibration process allows for gaze estimations to be compared with known ground truth values, since the locations of the calibration dots are pre-determined. With the assumption that the user is looking at each dot instead of somewhere else on-screen, gaze predictions are compared to the actual locations and metrics are calculated on the performance of the gaze detection system. Calibration is performed separately for each gaze detection system, allowing for comparisons to be drawn between the performance of the two systems. Another advantage of the calibration process is that the calibration results can be used to fine-tune the page-turning techniques. To give an example, if the gaze predictions from the iTracker gaze detection system had a consistent error in the y-axis such that each prediction was further down the screen than the actual point, this could be compensated for to ensure the page turns were triggered at the correct time.

3.1.1 Gaze detection systems

Two different gaze detection systems were implemented: the iTracker neural network and the SeeSo software development kit (SDK). The SeeSo SDK was developed by Visual Camp, a Korea-based company that focuses on eye tracking technology. Similarly to iTracker, SeeSo is an appearance-based gaze estimation system that uses the front-facing camera of a device to produce real-time gaze predictions (Visual Camp, 2021). SeeSo operates directly on a camera feed, whereas iTracker has specific inputs which must be
prepared before any gaze predictions are performed. To transform the front-facing camera frames into suitable inputs for iTracker, a face and eye segmentation module was implemented. Although the SeeSo SDK provides a possibility of doing user-specific calibration, it was decided not to utilise this calibration in order to fairly compare the two gaze detection systems.

3.1.2 Page-turning system designs

As mentioned in the introduction there are several possibilities of how to present new music to a user such as vertical or horizontal scrolling of the music, animated page turns and partially animated page turns. In an effort to evaluate a range of page-turning systems three such page-turning systems were implemented in the EyeGaze application: scrolling, single animation and double animation. The design decisions regarding the page-turning systems can be found below and the implementation details of how the gaze predictions trigger these systems can be found later in this chapter.

Figure 3.2: From left to right: Process of scrolling new page on-screen using scrolling page-turning system, with the new page highlighted in yellow for clarity.
Scrolling system

The scrolling system involves vertical scrolling of the music; one page of music is displayed at a time and as the user’s gaze progresses down the screen towards the bottom of the page, the page is scrolled upwards. New pages are appended below the current page, allowing new pages to be scrolled on-screen from the bottom. This process is illustrated in Figure 3.2. This system doesn’t contain explicit breaks between pages and this can lead to challenges when attempting to compare this system to the other animated systems.

Single animation page-turning system

The next page-turning system is the single animation system. This system displays a single page at a time, and when it detects that the user has reached the end of the page, it triggers an animation which turns to the next page of music. This animation is designed to imitate the appearance of a physical page being turned. A graphic demonstrating this process is given in Figure 3.3.

![Figure 3.3: Left to right: Bounding box for triggering animation shown in green, page turn animation starting after timer completes, peeling away current page to reveal next page.](image)

Double animation page-turning system

The third page-turning system is the double animation system, which initially triggers a partial page turn followed by a complete page turn. The rationale behind this page-turning system is that as a user is reading the last part of a page the next page can be partially revealed at the top of the screen, ensuring that when the user looks to the top left of the screen the correct music will be visible. This should ensure that the correct music is
always being displayed and that the section of music the user is looking at is always static. Each page turn consists of two animations, the first of which progresses from the top of the screen to halfway down the screen, peeling away the current page to reveal the next page of music underneath. Once it is identified that the user has finished reading the current page and is looking at the partially revealed next page, the second animation is triggered. This second animation continues where the first finished, scrolling down to the bottom of the screen, revealing the rest of the next page. Figure 3.4 depicts the various stages of this page-turning system, with the top row showing the first animation and the bottom row showing the second animation. The next page is highlighted solely for clarity of demonstration.

![Figure 3.4: Top left: Bounding box to trigger first animation shown in green. Top centre: Next page (highlighted) being revealed from top. Top right: First animation complete with half of the next page (highlighted) shown. Bounding box to trigger second animation shown in green. Bottom left: The rest of the next page being revealed. Bottom right: Second animation completion.](image-url)
3.2 Implementation of the EyeGaze application

3.2.1 Calibration

Upon opening the app, users are presented with an option to begin the calibration process. During the calibration ten dots are shown on-screen, one at a time. The design of the dots was influenced by the dots used in the GazeCapture application, with a small black dot surrounded by a larger red dot to encourage users to look at the centre of the dot (Khosla, 2016). The dots were presented to the user in an order that reflected how their gaze might move while reading a piece of sheet music - beginning in the top left and progressing left-to-right in a gradual downwards trajectory. Ten dots are presented altogether, one in each corner of the screen and six spread throughout the screen. The layout of the calibration dots is shown in Figure 3.5, although in the EyeGaze application only one dot is visible at a time. Each dot was displayed until 30 frames were captured where valid gaze detection could be performed. If no face was detected from the frame of the front-facing camera then no gaze detection would be performed and a message would be displayed to inform the user that no face was visible. At the end of the calibration process the 600 gaze predictions, along with the respective ground truths associated with those predictions and several key metrics would be written to a .CSV file and stored on the iPad.

![Figure 3.5](image.jpg)

Figure 3.5: Layout of dots used in calibration sequence, with each dot labelled in the order of appearance.
3.2.2 Face and Eye segmentation

The iTracker CNN requires four inputs: a cropped image of a face, a cropped image of each eye and a binary mask that indicates the face position in the original image. These inputs can be seen in Figure 3.6. To prepare these inputs several operations were performed on each frame of the front-facing camera. Face and landmark detection was performed first to identify if there were any faces present in the frame, using the Vision framework (Apple, 2017). If no faces are detected in the image, no further operations are performed and a flag is set to indicate that facial detection failed. The Vision facial detection returns a bounding box for each face found in the image, so the largest bounding box is used to crop the face out of the image, producing the first input for iTracker. The Vision framework also returns a series of points that correspond to facial landmarks; the distribution of these points can be seen in Figure 3.7.

Bounding boxes for each eye are generated from the facial landmark points, with a small buffer on all sides to include some of the area around the eye. This buffer was included because during the implementation it was observed that the system performed...
better with a wider crop of the eyes, presumably because the CNN was trained on wider eye crop images. After cropping the eyes from the original image, the only remaining input for iTracker is the binary mask which is constructed using the position of the detected face bounding box in the original image.

![Image showing facial landmarks]

Figure 3.7: *Left:* Revision 2 of the Facial Landmarks algorithm, giving 65 facial landmark points (lbsweek, 2018). *Right:* Revision 3 of the Facial Landmarks algorithm, giving 72 facial landmark points (lbsweek, 2018).

3.2.3 iTracker CNN

The iTracker CNN was originally implemented using the Caffe deep learning framework. In order to use the model on iOS, it was converted from a Caffe format to a CoreML format. Apple provides a coremltools package to assist with conversion from various model types to the CoreML format. To convert iTracker involved using a deprecated version of the coremltools package and rewriting parts of the deploy.prototxt file which describes the structure of the model, as converting from a Caffe format to a CoreML format is no longer supported. Once iTracker was integrated with the EyeGaze app, manual testing was required to validate that the inputs were in the correct format. This manual testing involved building a testing platform which was eventually reworked to become the user calibration process. During initial testing of the iTracker system it was found that the gaze predictions were clustered towards the centre of the screen. Using scaling factors to scale the x and y components of the gaze predictions resulted in far more accurate predictions. These scaling factors were initialised to default values and dynamically updated to minimise the dot error as the calibration process was completed.
To help reduce the errors of both the iTracker and SeeSo gaze detection systems an approach was taken such that gaze predictions which were outside the screen bounds but within a close range to the screen were clamped to the edges of the screen.

3.2.4 Implementation of scrolling page-turning system

The main challenge regarding the scrolling system was implementing the algorithm to adapt the scrolling speed given gaze predictions. The screen is conceptually divided into three sections of differing sizes, corresponding to three different scrolling speeds. These different sections of the screen can be seen in Figure 3.8. The gaze prediction data from both gaze detection systems was found to be noisy, so a rolling average was taken of the user's gaze using the last fifteen gaze predictions. Using this rolling average helped to smooth out the relatively noisy gaze predictions. If the rolling average changes from one section of the screen to another, the scrolling speed is gradually updated until it reaches the correct speed for the new section. This gradual update of the scrolling speed is designed to avoid jarring changes in speed which could be distracting to the user.

When the PDF file is first opened, scrolling is disabled and is only enabled once the user's gaze has dropped below a threshold; in this case the threshold is a quarter of the height of the screen. This threshold is shown in Figure 3.8 for clarity. This prevents the situation of the scrolling beginning as soon as the PDF file is opened, which could be disruptive and potentially accidentally scroll music off-screen.

Another consideration was how to handle the situation of the user looking away from the music. If no face is detected from the front-facing camera feed then no gaze predictions are supplied, and in this situation the scrolling speed remains constant, under the assumption that the user is still playing the piece, but has simply glanced away momentarily.

3.2.5 Implementation of single animation page-turning system

In a similar fashion to the scrolling system, a rolling average of the last fifteen gaze predictions is also used to smooth out the noisy gaze predictions for the single animation page-turning system. When this rolling average enters a bounding box in the bottom right
corner of the screen a timer is started, and on completion of the timer the page turn animation is triggered. A timer is used as a rough prediction of the length of time it will take the user to finish reading the music contained within the bounding box. The area which defines this bounding box is influenced by the calibration process a user completes upon opening the app. The ten\textsuperscript{th} calibration point is used as an estimate of what gaze prediction would indicate the user is looking at the bottom right corner of the screen, and the bounding box is constructed using this as an anchor point. Once a page turn has been triggered, there is a delay before another page turn can be triggered, to avoid a situation where several pages are turned in quick succession by accident.

Figure 3.8: Various scrolling speed zones shown in green, yellow and red. Initial threshold for starting scrolling upon opening file shown in blue.

Consideration was given to the direction of the page turn; initially the animation was configured to begin in the top left corner of the page and progress to the bottom right corner, with the intention to maximise the length of time the music in the bottom right corner was on-screen. It was found during the user study that this was confusing to users as it was quite different from a standard page turn. As a result the page turn was reconfigured to start from the bottom right and progress from right to left across the screen, as depicted in Figure 3.3.
3.2.6 Implementation of double animation page-turning system

This system also uses a rolling average of the last fifteen gaze predictions to smooth out noisy gaze predictions, and when this rolling average enters a bounding box in the bottom right corner of the screen the first animation is triggered. This bounding box is shown in Figure 3.4. The first animation will complete and the system will stay in this state until the rolling average enters a second bounding box at the top of the screen, indicating that the user has finished reading the first page and has begun reading from the top of the screen again. This second bounding box is also shown in Figure 3.4. The rolling average entering this second bounding box will trigger the second animation, peeling away the rest of the first page from the halfway point to the bottom of the screen. Similarly to the single animation system, after each animation is triggered there is a delay before the animation can be triggered again, to avoid several animations happening in quick succession. It should be noted that the bounding boxes for both animations from this page-turning system are larger than the bounding box from the single animation system. The timing of the animations is not as important as in the single animation system, which allows the bounding boxes to be larger.
4 Evaluation of the EyeGaze application

4.1 Evaluation approach

When designing and implementing the EyeGaze application the goal was to investigate whether gaze detection systems could be effectively applied to this use case in a real-world scenario. To effectively evaluate this, the application would need to be tested in a variety of situations with different subjects and varying lighting conditions. This necessitated a user study, where the performance of the application could be evaluated based on the interactions these users had with the application. There were several different aspects of the application to evaluate: the effectiveness of the page-turning systems, user preference between the various page-turning systems and the performance of the two different gaze detection systems. A page-turning system can be considered effective if the correct music is on-screen at all times. With this in mind evaluation metrics were designed to compare the page-turning systems’ ability to keep the correct music on-screen at the correct time. Another element to evaluate was whether users had a preference between the different page-turning systems; to understand this a questionnaire was given to users during the user study. The performance of the different gaze detection systems was also of interest and in order to evaluate this user data was gathered during the calibration process of the application, allowing quantitative metrics to be gathered.

4.1.1 User study

With the intention of testing the application in a variety of situations, it was decided that the user study would be conducted in a location of each subject’s choosing. There were a variety of factors identified which could potentially affect the accuracy of the gaze detection systems, such as glasses or long hair partially occluding subjects’ faces, varying lighting conditions and varying relative positioning of the subject and the iPad. The only restriction placed on subjects while using the application was that face masks such as those used to prevent the spread of COVID-19 could not be worn. The reason for this restriction was that Apple’s Vision framework performed very poorly at face detection
when a face mask was worn, which would lead to a dramatic decrease in gaze detection accuracy.

Each subject selected their own piece of music and played through their piece six times; using each of the three page-turning systems with each gaze detection system. The subjects were then asked to complete a questionnaire about their experience. This questionnaire contained questions to qualitatively evaluate each page-turning system independently as well as giving subjects the opportunity to rate the page-turning systems relative to each other. As previously mentioned, gaze predictions and their associated ground truth values were collected during the calibration process of the application to qualitatively assess the gaze detection systems. This allowed for analysis of the distribution of the gaze predictions. Finally, to assess the performance of the page-turning systems the researcher present during the user study would collect data on whether each page turn happened early, late or at the correct time.

4.1.2 Metrics

Measuring whether page turns were early, late or at the correct time was suitable for the single and double animation page-turning systems, but not for the scrolling system. In order to record a similar metric for the scrolling system the number of jumps ahead or gaps in the music were recorded. If the system was scrolling too fast then the music being played would be scrolled off-screen from the top of the screen, leading to the user having to skip ahead in the piece. Similarly if the system was scrolling too slowly then the user would reach the bottom of the screen before more music was scrolled on-screen from the bottom, causing a pause in the music as they waited for the scrolling system to catch up.

Although the performance of gaze detection systems is commonly measured using angular resolution in degrees, this metric is unsuitable for the EyeGaze application use case. A more suitable metric would be the distance from estimated gaze points to actual points on-screen. During the calibration process of the application, 30 gaze estimations are collected for each dot shown on-screen. The arithmetic mean of the distances in centimetres from each of these gaze estimation points to the respective ground truth points gives an indication of the error of the gaze detection system. If each individual gaze
estimation is expressed as follows: \( est_{nm} = \{ est_{nm} X, est_{nm} Y \} \), then all of the gaze estimations gathered for a particular calibration dot can be expressed as: \( est_p = \{ est_{p1}, ..., est_{p30} \} \), as there are 30 gaze estimations for each calibration dot. A particular calibration dot will be expressed as: \( gt_p = \{ gt_p X, gt_p Y \} \), giving the list of all calibration dots to be: \( gt = \{ gt_1, ..., gt_{10} \} \). The mean of the distances from all gaze estimations relevant to a particular calibration dot \( gt_p \) is given as:

\[
\text{doterror}_p = \frac{\sum_{m=1}^{30} \sqrt{(gt_p X - est_{pm} X)^2 + (gt_p Y - est_{pm} Y)^2}}{30}
\]

This allows for the doterror of all calibration dots to be given as:

\[
\text{doterror} = \frac{\sum_{p=1}^{10} \text{doterror}_p}{10}
\]

This metric is similar to the pix\_dist metric sometimes found in related literature, with the difference being that the distance function is measured in centimetres instead of pixels for the doterror metric (Kar & Corcoran, 2018). Another metric we will be using is the \( x \) axis skew, which will be the arithmetic mean of the \( x \) position distances from each gaze estimation point to the respective ground truth point. Similarly we will use a \( y \) axis skew metric which will measure the arithmetic mean of the \( y \) position distances from each gaze estimation point to the respective ground truth points. The \( x \) and \( y \) axis skew for a particular calibration point is given as:

\[
x \text{ axis skew}_p = \frac{\sum_{m=1}^{30} \sqrt{(gt_p X - est_{pm} X)^2}}{30}, y \text{ axis skew}_p = \frac{\sum_{m=1}^{30} \sqrt{(gt_p Y - est_{pm} Y)^2}}{30}
\]

Thus the \( x \) and \( y \) axis skew for all calibration points is given to be:

\[
x \text{ axis skew} = \frac{\sum_{p=1}^{10} x \text{ axis skew}_p}{10}, y \text{ axis skew} = \frac{\sum_{p=1}^{10} y \text{ axis skew}_p}{10}
\]
4.2 Results

Each page-turning system has slightly different metrics by which their performance is measured. The single and double animation page-turning systems are both judged by the number of successful, early or late page turns but for the double animation system the early and late page turns are further categorised by which specific animation was triggered early or late. This allows for more granular analysis of where the system is failing. The number of jumps ahead or pauses in the music are counted for the scrolling system, to give an indication of whether the system was scrolling too quickly or too slowly in general.

<table>
<thead>
<tr>
<th>Scrolling page-turning system</th>
<th>Total page turns</th>
<th>Total number of jumps</th>
<th>Total number of pauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeeSo</td>
<td>18</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td><em>iTracker</em></td>
<td>18</td>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.1: Performance of the scrolling page-turning system, detailing the total amount of jumps ahead in the music and pauses in the music across all user tests.

<table>
<thead>
<tr>
<th>Single animation page-turning system</th>
<th>Total page turns</th>
<th>Total successful page turns</th>
<th>Total early page turns</th>
<th>Total late page turns</th>
</tr>
</thead>
<tbody>
<tr>
<td>SeeSo</td>
<td>18</td>
<td>0</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td><em>iTracker</em></td>
<td>18</td>
<td>2</td>
<td>14</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 4.2: Performance of the single animation page-turning system, detailing the total number of successful, early and late page turns across all user tests.
**Double animation page-turning system**

<table>
<thead>
<tr>
<th></th>
<th>Total page turns</th>
<th>Total successful page turns</th>
<th>Total early page turns (1st animation)</th>
<th>Total late page turns (1st animation)</th>
<th>Total early page turns (2nd animation)</th>
<th>Total late page turns (2nd animation)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SeeSo</strong></td>
<td>18</td>
<td>11</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td><strong>iTracker</strong></td>
<td>18</td>
<td>8</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.3: Performance of the double animation page-turning system, detailing the total number of successful, early and late page turns. For early and late page turns the specific animation which was triggered early or late is specified.

The gaze estimation information from the calibration portions of the user tests also allow us to compare the performance of the two gaze detection systems. In Table 4.4 the SeeSo and iTracker systems are compared, using previously explained metrics of dot error, x axis skew, and y axis skew. Tables 4.5 and 4.6 show the performance of each gaze detection system per calibration point. The calibration point numbers correspond to the order in which the calibration points are displayed, as shown in Figure 3.5.

**Comparisons between gaze detection systems**

<table>
<thead>
<tr>
<th></th>
<th>SeeSo</th>
<th>iTracker</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average dot error (cm)</strong></td>
<td>5.5598</td>
<td>7.1518</td>
</tr>
<tr>
<td><strong>Average x axis skew (cm)</strong></td>
<td>0.2985</td>
<td>-1.2539</td>
</tr>
<tr>
<td><strong>Average y axis skew (cm)</strong></td>
<td>-0.3755</td>
<td>7.2189</td>
</tr>
</tbody>
</table>

Table 4.4: Overall comparisons between the two gaze detection systems.
SeeSo gaze detection performance per calibration point

<table>
<thead>
<tr>
<th>Calibration dot</th>
<th>Average dot error (cm)</th>
<th>Maximum dot error (cm)</th>
<th>Dot error range (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.583</td>
<td>10.718</td>
<td>8.577</td>
</tr>
<tr>
<td>2</td>
<td>5.300</td>
<td>11.156</td>
<td>8.448</td>
</tr>
<tr>
<td>3</td>
<td>7.025</td>
<td>9.732</td>
<td>5.178</td>
</tr>
<tr>
<td>4</td>
<td>3.550</td>
<td>8.016</td>
<td>6.667</td>
</tr>
<tr>
<td>5</td>
<td>4.221</td>
<td>7.700</td>
<td>5.885</td>
</tr>
<tr>
<td>6</td>
<td>6.162</td>
<td>7.732</td>
<td>4.478</td>
</tr>
<tr>
<td>7</td>
<td>3.918</td>
<td>8.481</td>
<td>6.907</td>
</tr>
<tr>
<td>8</td>
<td>4.421</td>
<td>11.435</td>
<td>9.358</td>
</tr>
<tr>
<td>9</td>
<td>8.846</td>
<td>10.841</td>
<td>3.735</td>
</tr>
<tr>
<td>10</td>
<td>6.571</td>
<td>12.005</td>
<td>9.649</td>
</tr>
</tbody>
</table>

Table 4.5: Metrics calculated for each calibration point across all user tests for the SeeSo gaze detection system.

iTracker gaze detection performance per calibration point

<table>
<thead>
<tr>
<th>Calibration dot</th>
<th>Average dot error (cm)</th>
<th>Maximum dot error (cm)</th>
<th>Dot error range (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.945</td>
<td>8.433</td>
<td>5.984</td>
</tr>
<tr>
<td>2</td>
<td>6.947</td>
<td>10.191</td>
<td>5.229</td>
</tr>
<tr>
<td>3</td>
<td>6.503</td>
<td>12.767</td>
<td>10.891</td>
</tr>
<tr>
<td>4</td>
<td>5.424</td>
<td>10.436</td>
<td>7.677</td>
</tr>
<tr>
<td>5</td>
<td>5.784</td>
<td>11.654</td>
<td>9.281</td>
</tr>
<tr>
<td>6</td>
<td>4.985</td>
<td>9.457</td>
<td>8.379</td>
</tr>
<tr>
<td>7</td>
<td>5.643</td>
<td>7.664</td>
<td>5.037</td>
</tr>
<tr>
<td>8</td>
<td>8.343</td>
<td>12.135</td>
<td>7.137</td>
</tr>
<tr>
<td>9</td>
<td>10.815</td>
<td>18.325</td>
<td>12.084</td>
</tr>
<tr>
<td>10</td>
<td>11.127</td>
<td>14.914</td>
<td>7.480</td>
</tr>
</tbody>
</table>

Table 4.6: Metrics calculated for each calibration point across all user tests for the iTracker gaze detection system.
4.3 Analysis of results

The first thing to note from the results of the user study is that it is clear from Tables 4.1, 4.2 and 4.3 that the three page-turning systems varied in performance. The single animation system performed dramatically worse than the double animation system, with far fewer successful page turns. The results from the questionnaire also reflected that users preferred the double animation system to the single animation system, with 83% of users ranking the double animation system above the single animation system. It is a bit more challenging to directly compare the scrolling system and the two animation systems, as they don’t share the concept of successful page turns. It is clear from the questionnaire results that the scrolling system was preferred by most users, with 66% of users ranking it as their most preferred page-turning system and no users ranking it as their least preferred. Figure 4.1 depicts user's experience while using each of the page-turning systems and it can be seen that user preference reflects the quantitative performance of the systems, with the scrolling system receiving the most positive response, followed by
the double animation system and the single animation system receiving the most negative response.

4.3.1 Failure cases in page-turning systems

Having established that there is a clear difference in the performance of the three page-turning systems, it is important to try and understand the underlying causes of the failure of the systems. Breaking down the unsuccessful page turns for the single animation system, 88% happen because the page turn was triggered too early. One of the possible causes for this is an imperfect implementation of the trigger mechanism. As described in Chapter 3, a rolling average of the last 15 gaze predictions was used to smooth out noise in the gaze predictions. This can cause unexpected results in edge cases.

One such edge case would be if the user begins by looking at the top left of the screen for some short period of time such that the rolling average gaze prediction is in the top left corner. If the user then glances below and to the right of the device screen this will cause new gaze predictions to be below and to the right of the screen. As the rolling average is updated, it will move from the top left corner of the screen in a relatively straight path to the user’s new point of regard, below and to the right of the screen. This rolling average will enter the bounding box in the lower right-hand corner of the screen as it is being updated, which can trigger a page turn even though the user never actually looked at the bottom right corner of the screen. Edge cases such as these were observed during the user study, where a pianist glancing at the positioning of their right hand regularly triggered an early page turn.

There are several ways that edge cases such as these could be mitigated: gaze predictions that lie far outside the bounds of the screen could be discarded or a check could be added that some number of the previous 15 gaze predictions lie within the bounding box for the trigger mechanism.

A more robust solution would be to identify the types of eye movement the gaze predictions describe. If there is a rapid and considerable change in the gaze predictions this is likely to be a saccade, an abrupt movement of the eye which dramatically changes
the subject’s point of regard. A saccade resulting in gaze predictions which are outside the bounds of the screen are likely to be caused by a musician glancing away from the music, and thus should not trigger page turns.

Early triggering of the page-turning system due to glances away from the music was also observed when users were using the double animation system, but did not cause as many issues for the scrolling system. One possible explanation for the scrolling system handling these cases is the gradual increases from one scrolling speed to another.

Another possible cause for the single animation system triggering page turns at the incorrect time could be the timer triggered when the rolling average of gaze predictions enters the bounding box in the bottom right hand corner of the screen. This timer was implemented with a constant time for simplicity, but the speed at which a user reads the sheet music varies and is dependent on the tempo of the piece as well as the formatting of the sheet music. A simple but perhaps more effective solution might be a dynamically changing timer with the duration of the timer being dependent on the length of time it takes the user to read the entire sheet of music. A more elegant solution could be to use computer vision techniques to identify the systems of music within the page and form a predictive model of where the user will be looking throughout the page. This model can then be updated with the gaze predictions, potentially yielding more accurate results. Such a system was proposed by Tabone et al. in their paper using Kalman filtering to create a gaze prediction model (Tabone et al., 2020).

The scrolling system caused more jumps in the music than pauses in the music, where jumps in the music would be caused by the system scrolling too fast. One of the possible reasons for this could be the design decision taken to only scroll the music in a single direction. A simple solution would be to allow scrolling of the music in both directions, such that if a user was looking at the top of the screen the music would scroll from top to bottom, revealing music from previous pages in the sheet music.
4.3.2 Gaze detection systems

Comparing the two different gaze detection systems it is apparent that SeeSo is generally more accurate, as shown by the lower dot error in Table 4.4. The increase in accuracy doesn’t appear to be correlated to an increase in page-turning performance though; SeeSo performs better than iTracker in the double animation system, worse in the single animation system and causes the same number of jumps or pauses in the music for the scrolling system. This might suggest that the implementation of a page-turning system matters more than the accuracy of the gaze detections system with regard to page-turning performance.

Table 4.4 also shows that the average x axis skew and average y axis skew were significantly larger for iTracker. This indicates that on average gaze predictions from the iTracker system were \(~\)1.25cm to the left and ~7.22cm lower than the ground truth locations. Part of the reason for these decreases in accuracy may have been due to the scaling constants introduced to improve the accuracy of the iTracker model.

Tables 4.5 and 4.6 allow for comparison of the accuracy of the two gaze detection systems for each calibration point. It is clear from these tables that there is quite a large difference in the accuracy of the gaze detection systems depending on where the user is looking on-screen. The centre-most calibration dots (dots 4 and 7) have much lower average dot errors compared with the dots in the bottom left and right corners (dots 9 and 10 respectively). For both gaze detection systems the average dot error of the calibration dot in the bottom right corner of the screen is nearly twice that of the centre-most calibration dots, and with a screen width of 18cm an average dot error of 11.13cm in the bottom right corner for the iTracker system is quite poor performance. This is significant as it is the bottom right hand corner of the screen which is used to trigger the single and double animation systems. Interestingly, these findings reflect those of Khosla et al. in their paper which introduced iTracker, where it was noted that as the distance from the point of regard to the camera increased the accuracy of the system decreased.
It should also be noted that the performance of both gaze detection systems may have suffered due to the unrestricted lighting conditions during the user study, as well as the unrestricted relative positioning of the user and the device. A lack of person-specific calibration or training of the two gaze detection systems could also be a contributing factor.

One possible solution to this decreased accuracy in an important area of the screen for page-turning would be to design the application to be used with the tablet device upside down. The front-facing camera would then be at the bottom of the screen, much closer to the bottom right hand corner of the screen and potentially resulting in increased accuracy in the gaze predictions in the bottom right hand corner.

4.3.3 Summary

The performance of the page-turning systems was varied and overall poorer than expected. There are multiple possible reasons for this including flawed implementation of the trigger mechanisms and a lack of fine-grained accuracy from the gaze detection systems. It should be noted that both the qualitative and quantitative performance of the page-turning systems is reflective of the required accuracy from the gaze detection systems. The scrolling system requires the least accuracy in gaze predictions, where the system only needs to identify which of several large on-screen regions the user’s gaze lies within. The double animation system requires more accuracy, as can be seen in the large bounding boxes used to trigger each animation in Figure 3.4. The single animation system required the most accuracy from the gaze detection systems, with the smallest bounding box for triggering animations of the three systems and correspondingly the worst performance of the three systems. Trends were identified in the distribution of gaze prediction errors, where a correlation was drawn between the on-screen location of a user’s point of regard and the accuracy of the gaze prediction. Additionally several possible improvements and mitigations were suggested for both the page-turning and gaze detection systems.
5 Conclusion and Future Works

5.1 Conclusion

This dissertation focuses on exploring the feasibility of using gaze detection to build an automated sheet music page-turning system for a tablet device. Two different gaze detection systems were used in the application and compared against each other. The results of these comparisons indicated that although there was a difference in accuracy between the two systems, this did not strongly correlate with a difference in page-turning performance. Three different page-turning systems were implemented in the application to identify differences in page-turning performance and user preference between the different systems. To effectively evaluate the various elements of the application in a general environment a user study was conducted, allowing for analysis of the qualitative and quantitative performance of the application. It was clearly observed that certain page-turning systems were more suited to gaze detection as a trigger mechanism for page-turning. Certain page-turning systems such as the scrolling system have inherently lower requirements on the accuracy of gaze predictions, leading to a better overall user experience. Although the performance of the gaze detection systems was poorer than expected, the results indicated that it is still possible to use these systems to build an automated page-turning application. Using person-specific calibration and training of the gaze detection systems, together with proposed improvements to the page-turning systems have the potential to result in a robust application capable of sufficient performance under varying conditions. Additionally, the responses to the questionnaire on user experiences indicated that there is strong interest in an application such as EyeGaze.

5.2 Future works

There are several areas that future research could focus on; a major such area is incorporating audio as an extra input in determining a user's position in the piece of music. This could improve the accuracy of the system in general, and would be particularly useful in certain situations. Advanced audio analysis might not be necessary in all situations; audio could be used as an indication that the user has started playing the piece of music simply by examining changes in audio levels. Another situation where audio would be useful is determining when a user has glanced away from the music and when the user has stopped playing the piece altogether. With the
current implementation of the EyeGaze application if no face is detected when a user is using the scrolling system then the scrolling continues at the same rate it was scrolling when a face was last detected. This implementation decision was based on the assumption that when no face is detected the user is momentarily glancing away from the screen to look at their instrument or other musicians and that the piece is still being played. The other possibility however is that the user has finished playing the piece and in this case the scrolling should stop. Using audio levels as an extra input to the page-turning system would allow situations like this to be handled correctly.

Another interesting problem that further research could focus on is handling repeats in the music. Sheet music occasionally contains sections of music that are repeated, which will cause the user to look at music that has already been displayed. This music might be further up the page being displayed, or it could require multiple backwards page turns. A similar situation could occur if a musician made a mistake while playing the piece and jumped backwards in the music to correct the mistake. A complete page-turning system would be able to identify and adapt to these situations.

During the analysis of the results in this paper it was identified that there were potential improvements in the handling of gaze predictions, where it was proposed that a more robust prediction model incorporating a reading model might yield better results. This would be an interesting improvement on the work outlined in this paper.
Bibliography


