University of Macau

Faculty of Science and Technology

MOOC-Gaze: Online Solution for Tracking Learners’ Gazing Dynamics of MOOC Videos

by

Kin, CHEANG KIN HENG, Student No: DB325355

Final Project Report submitted in partial fulfillment of the requirements of the Degree of Bachelor of Science in Computer Science

Project Supervisor

Mr. Miguel Gomes da Costa Junior

25 May 2017

Vol. 2
DECLARATION

I sincerely declare that:

1. I and my teammates are the sole authors of this report,
2. All the information contained in this report is certain and correct to the best of my knowledge,
3. I declare that the thesis here submitted is original except for the source materials explicitly acknowledged and that this thesis or parts of this thesis have not been previously submitted for the same degree or for a different degree, and
4. I also acknowledge that I am aware of the Rules on Handling Student Academic Dishonesty and the Regulations of the Student Discipline of the University of Macau.

Signature : _________________________
Name : Kin, CHEANG KIN HENG
Student No. : DB325355
Date : 25 May 2017
ACKNOWLEDGEMENTS

The author would like to express his utmost gratitude to UM for providing the opportunity to carry out a project as a partial fulfillment of the requirement for the degree of Bachelor of Science.

Throughout this project, the author was very fortunate to receive the guidance and encouragement from his supervisor Mr. Miguel Gomes da Costa Junior and Dr. Shirley, Weng In Siu.
ABSTRACT

Video-based lecturing has become the major teaching-and-learning activity of Massive Open Online Courses (MOOC). While students can learn any time at their own pace, course providers cannot have face-to-face communication with students to receive instantaneous feedback. Conventional ways like web forums allow learners to submit feedback after class, but this demands their additional effort to reflect on the lecturing process. Therefore, new techniques to evaluate learners’ in-class behaviour are needed. Gaze tracking is a technique which keeps track of the gaze position of a user. As a standard computer comes with a web camera, it is practical to perform non-intrusive gaze tracking of MOOC learners directly from user’s image. In this project, we created MOOC-Gaze, the software application to predict users' gaze positions of a MOOC video by analysing user’s facial and eye dynamics via web cameras. The prediction model is essentially a ridge-regression trained mapping function which takes users’ facial picture as input and outputs predicted gaze position as screen coordinates. The prediction model is built on-the-fly and user-specific, which requires one short calibration procedure before user starts watching the video. The predicted gaze positions can be visualized in heat-map form in a video replay for educators to observe learners’ individual and aggregated gaze patterns. This post-watch analysis can help in course evaluation, e.g. to find out whether gazing positions matching the expectation of the designer of the course video. MOOC-Gaze was designed to work with any kinds of video and to operate in a normal web browser. To improve the prediction accuracy, a literature survey of gaze tracking techniques was conducted and some solutions were proposed and tested. Our web framework together with the improved gaze prediction method delivers a new online gaze-tracking solution that can also be used in use cases other than MOOC.
# TABLE OF CONTENTS

## CHAPTER 1. INTRODUCTION  
1.1 System Overview  
1.2 Contributions  

## CHAPTER 2. LITERATURE SURVEY  
2.1 Gaze Estimation  
2.1.1 Model-Based Methods  
2.1.2 Appearance-Based Methods  
2.2 Gaze Feedback Analysis  

## CHAPTER 3. SYSTEM DESIGN  
3.1 System Design Overview  
3.1.1 Web-based Setting  
3.1.2 Use Cases  
3.2 MOOC Application  
3.3 Gaze Estimation Core  
3.3.1 WebGazer  
3.3.2 Eye Features Extraction  
3.3.3 Model Training  
3.4 Data Storage  
3.4.1 Application Data  
3.4.2 Gaze and Heat-map Data  
3.4.3 Experiment Data  
3.4.4 Consideration of Gaze Data Retrieval  
3.5 Gaze Feedback Analysis  
3.5.1 Heat-map Analysis  
3.5.2 Gaze Coverage Analysis  

## CHAPTER 4. SYSTEM IMPLEMENTATION  
4.1 Database  
4.1.1 MongoDB  
4.1.2 Embedding vs. Normalizing  
4.1.3 Scalability  
4.1.4 Retrieving Gaze Data  
4.1.5 Dealing with Large Document  
4.1.6 Video Storage  
4.2 Server Application  
4.2.1 NodeJS
LIST OF FIGURES

Figure 1: The MOOC-Gaze System Concept .............................................................. 10
Figure 2 Use Case Diagram ......................................................................................... 16
Figure 3 Workflow for Gaze Collection ...................................................................... 17
Figure 4 Comparison of Features Extractor using ClmTrackr and JSFeat ................. 18
Figure 5 Entity Relationship Diagram (ERD) ............................................................. 19
Figure 6 Different Areas in Gaze Coverage Analysis ................................................. 21
Figure 7 Embedding of Gaze Data ............................................................................... 23
Figure 8 APIs of server application ............................................................................. 25
Figure 8 Workflow of Heat-map Analysis .................................................................. 26
Figure 9 Workflow for Gaze Coverage Analysis ........................................................ 27
Figure 10 Batch Process of User Gaze Collection Documents .................................... 27
Figure 11 Workflow for Concurrent Workers to Build Heat-maps ............................. 28
Figure 12 MapReduce Process for Merging Heat-maps .............................................. 29
Figure 13 User Interface of MOOC-Gaze ................................................................. 30
Figure 14 Class Diagram of Client Application ......................................................... 31
Figure 15 Course Overview ......................................................................................... 32
Figure 16 Table of Contents ....................................................................................... 32
Figure 17 Calibration Screen ....................................................................................... 33
Figure 18 Course Video Player .................................................................................... 33
Figure 19 Heat-map Player .......................................................................................... 34
Figure 20 Table of Content for Viewing Reports ....................................................... 34
Figure 21 Gaze Coverage Report ............................................................................... 35
Figure 22 Heat-map for the First Screen in Video ....................................................... 36
Figure 23 Heat-map for the Second Screen in Video .................................................. 37
Figure 24 Heat-map for the Third Screen in Video ..................................................... 37
Figure 25 Heat-map for the Forth Screen in Video ...................................................... 38
Figure 26 Heat-map for the Fifth Screen in Video ...................................................... 38
Figure 27 Heat-map for the Sixth Screen in Video...................................................... 38
Figure 28 Report for Focused Student ....................................................................... 39
Figure 29 Report for Distracted Student .................................................................... 39
Figure 30 Focused Student: Gaze Coverage Inside AOI ........................................... 40
Figure 31 Focused Student: Gaze Coverage Outside AOI ......................................... 40
Figure 32 Distracted Student: Gaze Coverage Inside AOI ....................................... 40
Figure 33 Distracted Student: Gaze Coverage Outside AOI ...................................... 40
CHAPTER 1. INTRODUCTION

With the rise of MOOC (Massive Open Online Course) in education, learning via videos gradually gains its popularity. As the transition of live spot teaching, it’s important to have adequate techniques to evaluate students’ feedback towards these pre-recorded materials, e.g. their attentiveness distribution over the video duration, how much they understand on each part of the video lecture, etc.

Web forum has been a very general used platform to collect students’ feedback, which is used by many MOOC platforms like Coursera. However, this method applies only to those who are willing to ask questions or give answers, which may not be really helpful to relatively passive learners who only watch a video and leave. There have been studies analysing students’ behaviour in video learning by tracking user clickstreams on video-playing manipulation, e.g. fast forward, rewind, pause, and so on [15]. These natural actions for a video watcher may tell how a MOOC learner spend his focus on different sections of a video, his extent of engagement in the course material and even a prediction of his continuity on this web course [15]. The advantage of this method is that it does not require students’ additional effort to provide feedback nor to be interrupted while studying.

Inspired by click stream method, this project aims to compose another type of non-interruptive solution that helps capture MOOC learners’ reaction, which is by gaze estimation techniques, provided that web-cameras have been so widely spread in personal devices nowadays. Hence this project may contribute to MOOC learner behavioural studies together with the clickstream method mentioned above.

In addition, gaze estimation through a normal web camera also contributes to the evaluation of MOOC teaching and learning in many ways, e.g. the gaze path on a course slide of a student can become a useful feedback for the course material make, the common gaze focus of certain place in the video frame of a large group of students may tell educators unexpected information, the duration change of their gazing at the same area may tell information about their attention change pattern, etc. This technique is promising to facilitate remote studies. In fact, Gaze estimation is a long-discussed topic in human-machine interaction. This refers to the technique which estimate the exact region of a screen that a user is currently looking at, given his face/eyes image captured by camera. This project mainly focuses on the solution that does not require special device but only a general web-camera, which is supposed to be applied in MOOC classrooms.

The project aims to propose a practical solution, with implementation open-sourced to facilitate the public adoption of using gaze estimation techniques in MOOC learning. The project mainly contributes in two aspects: 1) To propose gaze estimation which is suitable for MOOC use cases; 2) To propose the application to discover information from gaze data to improve MOOC teaching. Although this project is inspired and designed mainly for MOOC use cases, other scenarios may also benefit from it, e.g. assistant human-computer communication for disabilities, live streaming app user interaction enhancements, and so on.
1.1 System Overview

This project aimed to build a web-based platform where course providers can put their resources and opened to the public. In addition to such common functionalities which current MOOC platforms already have, the project tried to incorporate a gaze tracking solution into platform to track the gazes of users while they are watching the course videos. The recorded gazes can be later visualized as a heatmap overlaid on each frame of a corresponding video’s replay. Course providers can also view the statistics of users’ gaze data on the platform. The functions serve as a way of feedback to help course providers better understand how the course video influence their online students’ behaviour by observing their gaze dynamics.

Below is an overview of how the platform operates to support course providers and students in MOOC learning.

![Figure 1: The MOOC-Gaze System Concept](image-url)
1.2 Contributions

In this project, we have the following major achievements:

- Applied gaze-tracking techniques in MOOC education field to provide a new way of student feedback, which the feedback data is not questionnaires, not forums, but gazes
- Implemented MOOC-Gaze platform, the first MOOC platform which can track users’ gazes via a normal web camera
- Applied heat-map technique as gazed position visualization in MOOC videos
- Improved the performance of a recent web-based gaze tracking package
CHAPTER 2. LITERATURE SURVEY

In order to track learners’ gaze dynamics and provide useful feedback to teachers, there are two main challenges we have encountered: gaze tracking and gaze feedback analysis. Gaze tracking provides the position in the video which learners are focusing on. With these positions, gaze feedback analysis provides useful feedback to teachers. In this chapter, we will summarize the related work in these two fields.

2.1 Gaze Estimation

In the literature, it is almost well-recognized that the approach towards gaze estimation is divided into two sets: model-based methods and appearance based methods.

2.1.1 Model-Based Methods

In model-based methods, features of face, head pose and eyes are carefully designed and extracted from a user-capture image with different kinds of image processing techniques. These features are extracted based on a carefully designed 2d/3d features model, e.g. a 2d eye model composed of eye lids and eye centers as features [1], a 3d head model including facial features and facial contour symmetry information as features [5]. These features are then map to screen coordinates/shifted angles compared to anchor position, by a special model/function, e.g. linear regression model, support vector machine trained model, or simply manually crafted formula-based model. Usually the work flow in a model-based solution goes through the following steps: facial feature extraction, eye-specific feature extraction, building mapping function and output gaze position. Precise eye-region localization in an image and the mapping function are the two critical steps in a gaze estimation solution.

Recently Sckodras proposed a feature-based solution which predict horizontal gaze direction and vertical gaze direction by two different mapping functions respectively, where eye centers and eyelids are key features of the prediction model [1]. In Skodras's solution, an user's image captured by the camera is firstly processed with a face detection algorithm proposed by Viola and Jones [2]. Then, a heuristic approach is adopted to locate the rough region of eyes. The regions of eye centers are then identified by a method which utilize symmetrical characteristics and chrominance information of the rough regions. After having the exact pixels of eye center, it continues to locate the upward eye lid and downward eye lid, which is latter used for determining the openness of the eyes. The solution also identifies a block of pixels, which is an image patch containing eye border and eye brown border. The center of the image patch is the reference point to calculate the displacement of eye centers and eye lids in frame sequences. Given the displacement distances and known gaze points in calibration phrase, two linear regression models are trained to predict horizontal and vertical gaze directions respectively. The displacement features of eye centers are used in horizontal model while the displacement features of both eye centers and eye lids are used in vertical model. The calibration duration of this solution lasts for a few seconds for a user. The solution obtains a less than 2-degree error in testing with Columbia gaze dataset.

Lin's proposed solution particularly addresses gaze estimation obstacles caused by illuminance variety of user environments [3]. After detecting the face with Viola-Jones's method [4], it processes the image with its proposed light filtering technique,
including gray-scale process and some customized color filtering functions, which turns out to be effective in removing the lighting noise in its experiments. After light filtering, the face image pixels are grouped into components by pixel connectivity as candidates for eye regions. The eye regions are selected based on a synergy of heuristic rules, symmetry characteristics and the relative distance between an eye and the face center. After locating the eye regions, pixels of irises are found by the following method: selecting pixels that is under 10% of Y value of the YCbCr color space of its histogram-equalized eye regions. A feature vector composed of 40 selected Fourier descriptors of two eye regions and irises are then trained with support vector machine given known gazing directions in calibration phrase. Lin uses his own source of data, which contains over 10000 images, in which about 3/10 the subjects wear glasses. It reports 90% of eye detection accuracy and gaze estimation accuracy in testing its dataset.

Heyman's solution purely depends on model-based mathematical calculation without the need of machine learning in its prediction step [5]. This solution takes head poses into consideration. It also starts its work by face detection with Viola-Jones algorithm [6]. It then tracks the detected facial features based on the normalized sum of squared differences (NSSD) with template matching technique to calculate the displacement vectors of facial features in user head movements. It also uses a Bayes's classifier to locate pixels of irises. The prediction model uses head pose information to calculate a rough direction and then fine-tune the result with eyeball orientation. In details, it builds up an initial frontal face model at the stage of facial feature detection. Based on this initial model, a series of computations are carried out to get the yaw angle, pitch angle and rotational angle of the head and eyeballs with displacement vectors, i.e. to "re-project" the user face onto a "cylindrical head and spherical eyeball model." The eyeball yaw angle and eyeball pitch angle are the final output of the horizontal direction and the vertical direction of this prediction model, respectively. It reports 7.09 degree mean error at horizontal direction and 4.4 degree mean error at vertical direction prediction using Columbia gaze dataset.

2.1.2 Appearance-Based Methods

In appearance based methods, the photometric appearance is directly used as input to estimate gaze. These methods capture other objects not limited to the eyes subject. With enough training data, other environment factors can be learnt into the model, which gives the potential capability to cope with variant conditions like illumination variation, head pose, quality of images etc. These methods allow gaze estimation technologies to be used in normal unconstraint conditions.

Appearance based methods typically require larger amounts of training data than model based methods, and the result closely relies on the training dataset. In the publishes, there has been several public dataset which is related to our system:

MPIIGaze dataset, proposed by Zhang [7], contains 15 subjects with total 213,659 images. The dataset is collected in variant size of laptops. Moreover, the sample are freely distributed over different daytime over more than three months, so to simulate in-the-wild setting. Zhang also proposed algorithm based on MPIIGaze dataset, the pre-processing employs Li et al.’s SURF cascade face-detection method [8] and Baltrušaitis et al.’s constrained local mode framework to locate facial landmarks [9]. Then estimates 3D head pose by using 3D facial shape model and normalizes the image in training space to get a centralized position and fixed distance. After pre-processing, the model
is trained by using multimodal convolutional neural networks (CNN). MPIIGaze dataset with the proposed algorithm can achieve accuracy of 13.9 degrees. However, unlike most of other public datasets, we found that MPIIGaze dataset contains only the normalized eyes images. In the MOOC use case, since the input images need pre-processing before predicting the gaze position by the model, the pre-processing especially 3D reasoning is not ideal to running in web browsers. Moreover, the achieved accuracy is limited.

TabletGaze Dataset proposed by Huang et al. [11] contains 51 subjects, and provides total 816 video sequences instead of images. The dataset is collected in 4 predefined postures: standing, sitting, slouching, and lying, but has no constraint for how they hold the tablet. The dataset is collected in single tablet (Samsung Galaxy Tab S 10.5). Based on Tablet Dataset, the algorithm performs the pre-processing and normalization so that the valid images can be extracted and has the comparable scale. Then eye-image cropping is performed by using a cascade eye detector [12]. Then the 5 features including contrast normalized pixel intensities, LoG, LBP, HoG, mHoG are extracted. Finally Use Random Forest with 100 trees to train the model, so to take advantage of the strong performance for feasibility in scaling to large dataset. By this method, in different environment settings, the experiment shows the mean error 3.17cm in person independent estimation, mean error of 2.5cm in person dependent estimation. However, there is no evaluation of estimation across different datasets, which bears the risk of significant dataset bias [13, 14]. Moreover, TabletGaze Dataset works in tablet settings rather than laptop settings. To work across devices, this method relies on transfer learning approach which gives difficulty to use in our project.

GazeCapture dataset proposed by Krafka et al. [14] contains 1474 subjects, 2,445,504 images, which is about 30 times as many participants as other datasets. The dataset is established by using mobile phones and tablets, which is scalable toward crowdsourcing approaches to produce large size and variability. The proposed algorithm iTracker for gaze estimation is relatively simple, it mainly uses convolutional neural networks (CNN), so to make best use of the large dataset. iTracker employs two characteristics to increase robust capability toward poor-quality eye detection: First, it does not rely on manual engineered system for head pose detection and normalization. Second, it takes two of more discriminative regions for prediction, including an image crop of the face together with its location in the image (termed face grid), as well as an image crop of tight region of the eyes. In order to support mobile devices usage with limited resources, the model complexity is reduced by lowering the image quality of other non-discriminative region, also by combining using a full model and a reduced model for estimation. The method can achieve prediction error of 1.71cm and 2.53cm without calibration on mobile phones and tablets respectively, and 1.34cm and 2.12cm with calibration on mobile phones and tablets respectively. Proposed experiment also shows it generalizes well over other datasets, with error of 2.58cm. The prediction time can be as fast as 0.05s for reduced model, with Apple face detection pipeline, overall can achieve 10-15fps. GazeCapture with iTracker overall achieve a good result. However, GazeCapture dataset is only available for permitted access at that time when our work is completed.

Although appearance based methods are believed to work well in normal unconstraint condition, and have potential to give acceptable accuracy, the several public datasets we have found neither have no permission to access nor have their own limitation which
we have mentioned above, such as performance, variability over devices etc. On the other hand, building our own dataset is costly thus impractical to our project. As a result, we tend to use model based method in our system.

2.2 Gaze Feedback Analysis

Gaze feedback are useful in many ways for evaluating teaching and learning of MOOC video, researchers are finding meaning of gaze pattern in terms of learning outcomes. Kshitij et al. [16] found the relationship between gaze coverage and learning performance. In their experiment, Areas of Interests (AOIs) are predefined in every slide in the video, which are visual blocks of content in the slide. 40 participants are asked to watch the video, then finished a post-test after watching the video. The measurement of learning performance is based on the post-test result. They found that students with good learning performance have the larger gaze coverage area (measured by the number of Attention Points) while students with bad learning performance have smaller gaze coverage area. They also found that students with good learning performance have less AOI missed while students with bad learning performance have more AOI missed.
CHAPTER 3. SYSTEM DESIGN

3.1 System Design Overview

Our system aims to make use of our gaze estimation and gaze feedback analysis technologies to improve MOOC teaching. Nowadays famous MOOC platforms like Coursera and edX are rapidly growing for over ten million people using. Many teachers produce their courses on those platforms attract students to watch. Although there exist many ways to deliver content, video is the mainstream way for teaching and learning. With our gaze estimation and analysis technologies extend to MOOC, feedback can be recorded while watching video, we believe that insights from the analysis can help teachers to improve teaching.

3.1.1 Web-based Setting

Our system is designed for a web-based setting, like most of the MOOC platforms. Web-based setting gives convenience to teachers and students for access through web browser with minimum setup. However, web-based setting has natural resource constraint, this factor should be considered when designing our application, especially for gaze estimation which will be discussed.

3.1.2 Use Cases

There are mainly two types of users in our system, they are teachers and students. While course video is the mainstream way to deliver content, our system focuses on this core functionality of a MOOC application. Students can browse video list, then choose for specific video to watch. Teachers, who create courses, should be able to browse video list and watch the video to review their content. Moreover, teachers can browse for several provided reports about students' learning feedback. Figure 2 shows the fundamental use cases of our system.

![Use Case Diagram](image)

*Figure 2 Use Case Diagram*

With above design idea of our system, in our design, several criteria need to be considered during our design. The system design should be usable in web-based settings. Accuracy is emphasized in data collection as well as analysis.
3.2 MOOC Application

In order to provide the system functionalities, our system needs underlying MOOC application. The MOOC application design in our system allows students to watch videos, which is the core functionality of other MOOC application as well. Specifically, our MOOC application provides several reports to teachers about students’ learning.

![Workflow for Gaze Collection](image)

*Figure 3 Workflow for Gaze Collection*

When student watches video, student browses for list of videos, chooses specific video to watch. Since our solution use model-based gaze estimation, student is first required to do the calibration for a series of find-and-click actions. During students watching, the gaze collection core will capture the webcam image and predict the corresponding gaze position. The gaze estimation is ongoing and is recording in the background. When finished watching, the gaze data will submit to server, then store to database for further analysis. Gaze feedback reports after analysis are available for teachers. Teachers can check reports periodically to get insight.

3.3 Gaze Estimation Core

In our system, gaze estimation core is responsible for providing current location of the student’s gaze, telling which visual object on the screen students are concentrating. Since output of gaze estimation will be used for analysis to provide useful learning feedback for teachers, accuracy of the feedback highly depends on the accuracy of gaze estimation. Hence accurate result of gaze estimation is required by our system. Moreover, gaze estimation is performed at a web-based environment, which is the nature of MOOC application, constraints on system resources need to be considered. For fulfilling above requirements, as a result, a new gaze estimation core has been designed.

3.3.1 WebGazer

Our core is designed bases on WebGazer [17] to provide gaze estimation functionality. WebGazer uses model-based method to achieve instantaneously gaze estimation. It is fully implemented in JavaScript and is capable to locally run in the client web browser, which is suitable to the web-based design in our system.

WebGazer can be functionally divided in three components: 1. the WebGazer framework; 2. Eyes features extractor; 3. ridge regression model. The WebGazer framework coordinates the calibration and the prediction. It gets the image input periodically, pass it to other components to achieve gaze estimation. In the calibration, it is also responsible for getting the click position to add a sample to training data. The eye features extractor is responsible for extracting the eye features from the original image. In WebGazer, the features extractor is referring a third-party face landmark detection library – ClmTrackr [18]. While the face landmark is detected each time, WebGazer gets the eyes landmark and perform some image filtering techniques to provide the input for the model. The ridge regression model use the eyes features as input to predict the screen coordinates of gaze estimation as output. Overall, WebGazer
can achieve each gaze estimation at around 100ms. However, through our evaluation, we found that WebGazer cannot assure an acceptable accuracy in variant people. The accuracy can be poor or good based among different people. Furthermore, we found that the unreliability was due to the features extractor.

In our small-scale evaluation, we found that in most of the case ClmTrackr fails to detect, especially for the sample who wear glasses. In a more comprehensive evaluation using public dataset, result shows that ClmTrackr achieve a very low success rate of 6%. For the detailed experiment of the result of ClmTrackr, please refer to report Vol.I Chapter 8.

### 3.3.2 Eye Features Extraction

In order to improve the reliability of eyes features extraction, we based on JSFeat HAAR Object Detector [19] to implementation our own feature extractor. Instead of extracting the small eyes region extracted by ClmTrackr, JSFeat’s detector can only extract at the face level. Our solution employs heuristic approach to crop the eyes region, is relatively constant to the face. However, compared to ClmTrackr, JSFeat’s detector gives larger eyes region around the eyes, which gives difficulty in achieving same accuracy of gaze estimation. On the other hand, JSFeat’s detector is more robust to people facial difference (e.g. wearing glasses), and this gives better usability for our MOOC application.

![Figure 4 Comparison of Features Extractor using ClmTrackr and JSFeat](image)

### 3.3.3 Model Training

With the larger eyes region, since objects other than eyes subject will be capture, it is supposed to give less accuracy in certain case of estimation, but will give higher reliability. In order to improve the accuracy to an acceptable level, our model employs idea from appearance-based method, which reflects in our calibration stage design. First, we increase the size of training data needed in calibration, which gives tolerance to other noises. Second, although most of the model-based method encourages users to fix their head pose, we encourage users to naturally move their head pose rather than fix head pose. Third, dots for user to click are averagely distributed and randomly appeared. Compared to appearing in order, this design can encourage users to naturally move their head pose, which can provide enough variability to training data.

### 3.4 Data Storage

Data storage is an important component in our system. Good database design can correctly store data, simplify system implementation. Also, the analysis tasks depend on a good database design to effectively retrieve data. Our system need to store lots of important data, such as application configurations, video information, gaze data, heat-
maps, analysis reports and experiment results etc. These data support application operations, gaze feedback analysis, generating analysis reports and our own project experiments. See Figure 5, an Entity-Relationship Diagram (ERD) shows the necessary entities in the storage, as well as their relationships. Note that the database design has not been normalized which will be discussed in Section 3.4.4.

![Figure 5 Entity Relationship Diagram (ERD)](image)

**Figure 5 Entity Relationship Diagram (ERD)**

*Green: Application Data; Blue: Gaze and Heat-map Data; Yellow: Experiment Data*

### 3.4.1 Application Data

Application data support the application operation need. The core functionality of the MOOC application need the following entities:

- **Config Entity**: stores the application configurations, with the configuration name and the value.
- **Videos Entity**: stores videos information, including video ID, video name, video size (width and height), URL, video length, and analysis configuration (e.g. heat-map quality).

### 3.4.2 Gaze and Heat-map Data

Gaze and heat-map data support gaze feedback analysis. Gaze data are obtained from gaze collection. Heat-map data are generated from gaze data. Here we focus on the entities needed to store, detail for gaze feedback analysis will be discussed. These data include following entities:
• **User Gaze Collection Entity**: stores the uploaded gaze collection for a video of single user. Note that gaze collection contains gaze of all frames of a video. Attributes include video ID, create date, and the gaze collection (which is list composed of the gaze of every frames).

• **Merged Gaze Frames Entity**: stores the merged gaze frame of all users, which is necessary for generating heat-map report. Note that the gaze frame contains only gaze of single frame of a video. Attributes include video ID, time (in second, 1~total length of the video), and gaze frame.

• **User Heat-map Entity**: stores the heat-map of single user of single frame in video. Attributes include video ID, time (in second, 1~total length of the video), reference to user gaze collection, heat-map, and create date.

• **Merged Heat-map Entity**: store the merged heat-map of all users of single frame in video. Attributes include video ID, time (in second, 1~total length of video), heat-map, and the size of total population.

• **Coverage Report**: stores the analysis reports which is ready for view. Attributes include video ID, and the content of report.

### 3.4.3 Experiment Data

We have performed several experiments thus data are needed to store as following:

• **Calibration Entity**: stores the data need in calibration error experiment.

### 3.4.4 Consideration of Gaze Data Retrieval

In the MOOC application use case, it is common to have massive of videos and online learners. Rather than traditional focus on transactional processing, the nature of gaze data focuses on analytical processing. The gaze data have the following characteristics:

• Gaze data has large volume depends on length of video and number of audiences;
• Gaze data are analytical rather than transactional;
• Gaze data are appending rather than updating;
• Gaze data are ordered by time;
• The output of gaze data analysis is merged (i.e. aggregated).

### 3.5 Gaze Feedback Analysis

In order to evaluate teaching and learning in MOOC video, teachers can check gaze feedback reports for insight in MOOC application. Our system provides several reports for teachers: 1. Heat-map analysis report; 2. Gaze coverage inside AOI; 3. Gaze coverage outside AOI.

In our design, we believe three reports together can provide some insights to teachers. Gaze coverages analysis reports can provide quick information of students’ learning feedbacks over time. Heat-map analysis report can provide detail visualization of students’ gaze area. These are available features for teachers discover their teaching strategies or probable improvement need.
3.5.1 Heat-map Analysis

In the heat-map analysis report, distribution of aggregated gaze position is represented in heat-map, and can tells teacher the attention level, the reading pattern, the potion in slide which students concentrate. Teachers can replay the video for the detailed heat-maps varying over time. For an example use case, if an important chart is missed by most of the students, the heat-map color can reflect gaze attention. Based on this insight, adjustment to the slide can be made.

3.5.2 Gaze Coverage Analysis

Study founds the relationship of gaze coverage and learning outcomes. Kshitij’s work suggests that students with better learning performance have larger gaze coverage area and less AOI misses. Gaze coverage analysis can tell teacher students’ attention level, the potential area which most of students missed. Gaze coverage analysis is good for teachers to overview the learning feedback over time.

3.5.2.1 AOI

In Kshitij’s work [16], Areas of Interest (AOI) are predefined areas which are expected for student’s attention, which we have discussed in Section 2.2. Our design use the concept of AOI. However, instead of human defined, when teacher watches the video, the gaze dynamics will be recorded as AOI.

![Figure 6 Different Areas in Gaze Coverage Analysis](image)

(i) Area with Teacher’s Gaze; (ii) Area with Only Teacher’s Gaze; (iii) Area with Student’s Gaze; (iv) Area with Only Student’s Gaze

3.5.2.2 Gaze Coverage Inside AOI

In the report for gaze coverage of AOI, students’ aggregated heat-map will be compared to heat-map of teacher’s watch, which defines AOI.

AOI Missed is define as the area with only teacher’s gaze over all area with teacher’s gaze:
\[
AOI\ Missed\ (%) = \frac{\text{Area with Teacher's Gaze} \setminus \text{Area with Student's Gaze}}{\text{Area with Teacher's Gaze}}
\]

In contrast, the Gaze Coverage Inside AOI is defined as following:

\[
Gaze\ Coverage\ Inside\ AOI\ (%) = 1 - AOI\ Missed\ Percentage
\]

Report will show the coverage in percentage, and show its variance over time in the chart. This chart also implies the learning outcomes of students over time. The higher coverage percentage means less AOI misses of students’ watching. On the other hand, the lower coverage percentage means the more AOI misses of students’ watching.

3.5.2.3 Gaze Coverage Outside AOI

In the report for extra gaze from AOI, students’ aggregated heat-map will also be compared to teacher’s heat-map.

\[
Gaze\ Coverage\ Outside\ AOI\ (%) = \frac{\text{Area with Student's Gaze} \setminus \text{Area with Teacher's Gaze}}{\text{Area with Student's Gaze}}
\]

Report will show the area of extra region that students watch in percentage. This report is designed to shows teacher for the unexpected concentrations, which are probably distractions in students’ learning.
CHAPTER 4. SYSTEM IMPLEMENTATION

4.1 Database
Our system is implemented based on non-relational database to store the massive gaze data and retrieval for analysis. Besides, commonly used relational databases emphasize on normalization, retrieving gaze data will involves a number of complex queries and join operations, lead to poor performance which will not fulfil the criteria of our system design. These factors drive us using document oriented database in our implementation.

4.1.1 MongoDB
MongoDB is suitable for our implementation. As a document oriented database, the data documents are store in format similar to JSON, and provide convenience to manipulate like native object. Like general document oriented database as well as NoSQL database, MongoDB also provides good scalability, worked together with our scalable NodeJS server, our system gives potential to handle massive scale of MOOC application. More importantly, MongoDB provides great advantage on our analysis tasks. The powerful MapReduce and aggregation query of MongoDB allows our system to base on to provide analysis function on massive gaze data.

4.1.2 Embedding vs. Normalizing
Our implementation make use of embedding. The embedding support of document oriented database, enables the gaze data to be embedded in a document. Compared to traditional relational database emphasis normalization, embedding usage avoid unnecessarily joining on large data collection. For example, for the User Gaze Collection Entity (which we discussed in Section 3.4), the gaze series, composed of different values over time, can be embedded as a list, rather than breaking into tuples. When a user gaze collection is analysis, the gaze data can be returned within the document instantly in our database design, rather than complex joining which cannot finished in constant time.

```
{  
  "id": ObjectId("591b2a5a01d25560d8106f"),
  "video_id": 0,  
  "user_id": 1,
  "create_date": ISODate("2017-05-11T13:37:00.000Z"),
  "gazes": [
    { 
      "time": 0,
      "gaze": {
        "x": 1134.59158415842,
        "y": 20.88828898981405
      }
    },
    { 
      "time": 0.1,
      "gaze": {
        "x": 129.3336383363317,
        "y": 31.6659292491749
      }
    },
    ...
  ]
}
```

*Figure 7 Embedding of Gaze Data*
4.1.3 Scalability
MongoDB comes with good scalability in a distributed system, the analysis functions MapReduce and Aggregation pipeline enables distributed computation. In our deployment, we based on a cloud platform mLab, which gives easy options to scale out.

4.1.4 Retrieving Gaze Data
Our implementation use efficient practice to retrieve gaze data. As we discussed in Section 3.4, the two main types of gaze data entity are gaze frame and gaze collection. Gaze frame refers to the gaze data for single frame in video. Gaze collection refers to the gaze data series for the whole video. In our implementation, most of the retrieval is through indexing. The gaze frames are accessed by video ID and time, and the gaze collection are accessed by video ID, they can hit the index thus ensure efficiency.

4.1.5 Dealing with Large Document
There is needed to store the heat-map in database, however, the file size has limitation on 16MB by default. For example, a video of resolution 1024*768, there are 786,432 pixels, that means 786,432 values (of Int32) needed to be stored in database, it is around ~25MB of size, exceed the upper limit 16MB of MongoDB. In the design of MongoDB, the large document gives difficulty on updating index. Moreover, accessing such document will take over seconds of time. For the solution, the resolution of heat-map is reduced to 100*50. For analysis purpose, this approach without compromising much accuracy of result can ensure the efficiency. In fact, the heat-map size can be configured for customization in the database for every video specifically.

4.1.6 Video Storage
Most of our data, including application data, gaze data and experiment data are stored in the database. However, course videos are large in size, when students watch videos, the traffic load of accessing will be obvious. Instead of database, course videos and large course materials are stored in cloud platform Amazon S3. Amazon S3 provides high throughput and availability, supports multiple accessing loads. While the responsibilities of each part are clear, the implementation gets simplified.

4.2 Server Application
Server lays in the middle layer of our system between database and client. Server handles client’s requests, communicate with database. In our system, however, server layer also coordinates the analysis job.

4.2.1 NodeJS
We use NodeJS for server implementation. It is suitable for lightweight developing, which save most of our unnecessary workloads. It reflects in minimizing lines of server code, the number of code can keep to around 670 lines, while 440 lines were for the report building function which is compose of relatively complex logic. The simplified logic of our code can give good readability, good quality from errors and bugs and other lots of advantages.
4.2.2 Server APIs

Server communicates with the client through server APIs, since our client use a fat-client design. This independency can simplify our implementation. The responsibilities of APIs of the server are quite strict-forward, see Figure 8. First, the server supports fetching of necessary application data from database, such as videos information. Second, the server support gaze data collection. Gaze data are handled through the server layer to database. Third, the server coordinate the gaze feedback analysis job.

![Figure 8 APIs of server application](image)

4.2.3 Dealing with Concurrency

The server application can run concurrently, when retrieving many documents for a request (e.g. several gaze frames for visualization use), the response should wait for all sub-jobs completed then return synchronously. In implementation, many call-back functions are embedded into another call-back functions as a result, this cause “call-back hell”.

Using events or promises can solve this problem. For events, when sub-worker is created, the worker counter is registered. When worker’s job finished, it unregistered from the counter and issue a finish event trigger. Every finish event signal received by the event, it checks the counter. Only when the counter shows that all worker’s jobs are finished, the main job finishes and returns. However, the logics of events is quite complicated, in our later implementation, we use promises instead. Through a series of .then() and .all() statements, the sequential or parallel logic can be implemented like waterfalls.

4.2.4 Scalability

NodeJS has good scalability. Considering the massive scale of typical MOOC application, in our implementation, NodeJS instance can be easily scale out. In fact, the deployment of our application is based on Heroku cloud platform, NodeJS instances run on dyno workers which supports scaling.

While the server application and database solution is scalable for massive scale, with our system design, the data analysis becomes possible toward large set of gaze data.
4.3 Gaze Feedback Analysis

4.3.1 Heat-map Analysis

Heat-map analysis aims to generate heat-map visualization based on gaze data. As our early implementation, the heat-map generation is performed on client side, based on a JavaScript library “heatmap.js”.

4.3.1.1 Process

A number of heat-maps need to be generated for every frame in a video. For every heat-map, the gaze data of such frame need to be ready then use “heatmap.js” setData() method to generate.

Since client application submits user gaze collection, in order to provide the merged gaze data for every frame, all related user gaze collection need to be proceed which is shown in Figure 9. First, every user gaze collection is split into user gaze frame (i.e. single frame for user gaze collection). Second, every user gaze frame need to add into merged gaze frame. Finally, the merged gaze frame provides input for heat-map generation.

4.3.1.2 Arrival Process

As we discussed, the volume of gaze data depends on the length of video also the number of users, which typically will be large. When user need to view heat-maps for specific video, the processing will take too long time.

Instead of processing when server receives request, it starts when the user gaze collection first arrives. The server update the merged gaze frames incrementally when new user gaze collection is submitted. When gaze frames data are needed for heat-map generation, they can be directly return without further process need.

4.3.1.3 Limitation and Further Improvement

For each gaze data arrival to server, the merged gaze frames documents need to be updated. When multiple arrivals, the adding will be inefficient due to the locking of the documents for updating. While MongoDB provide $push operator that could handle this conflict, some level of efficiency is still compromised. Thus, in our later development in gaze coverage analysis, the merge operation is performed by batch, which could solve this problem.
The heat-map generation is put in the client. The merged gaze frames are needed for generation, which is composed of all user gaze frames and typically large. This could be impossible when gaze data grow. In our later implementation in gaze coverage analysis, we have implemented solution to base on NodeJS server to generate heat-map, and directly stored into database. When requested, the aggregated heat-map can be directly returned to users. For the above two limitations, future improvement to the heat-map analysis is possible.

### 4.3.2 Gaze Coverage Analysis

The gaze coverage analysis is performed by the server and the database. Server can coordinate steps of the analysis, and do some more complex work like heat-map generation. Database can perform some data processing, by using MapReduce and Aggregation pipeline in MongoDB, the processing in database is closer to data thus can provide better efficiency, the temp tables can also be stored in database. Our implementation uses both server and database, and employs the advantage for both.

#### 4.3.2.1 Batch Process

In our implementation, the user gaze collection documents is processed by batch. The program can start by calling build report interface in server, it can also run periodically by integrated with server's timer. For the program, it will check the last_generated_heatmap and last_aggregated_heatmap values in database for the last run time. The documents after the last run time will be processed by batch, then the gaze coverage analysis will perform and the reports will be updated.

#### 4.3.2.2 Initialization

The initialization step focuses on preparation before the analysis starts at the server side. At server side, some data including videos data, configuration data are needed along the analysis process, so these data should be loaded from database in advance. The context in our app will store these necessary data for future use.

Context will serve the data which will be used among all the steps of analysis with many concurrent workers, this could reduce the transportation between database. When a sub-worker is created, a copy of the context will be passed to it.
4.3.2.3 Building Heat-map

The heat-map building step converts the gaze data of every single user gaze collection into heat-map. In fact, the heat-map rendering is performed on the canvas provided by NodeJS, this requires graphical computation resources thus put it in the server layer rather than the pipeline database layer. The heat-map building is performed on concurrent workers, this could improve performance and reduce database access overhead. Figure 12 shows the workflow of heat-map generation.

![Figure 12 Workflow for Concurrent Workers to Build Heat-maps](image)

The heat-map generation starts from querying gaze data from database. For each user gaze collection arrives, the server creates a worker for generating heat-map. To generate heat-map, the gaze data is rendered in the canvas by using the JavaScript library “Heatmap.js”. The colour of heat-map is set to white, while the meaningful part of data will be the alpha value in the RGBA of each pixel. Then the values of pixels can be saved to database in the collection “user heatmaps” and finish the work. At the end, the main program collects all finish signals from sub-workers and the step will be finished. The coordination of concurrent workers is achieved by the use of promise, which we will discuss afterward.

4.3.2.4 Merging Heat-map

The heat-map merging step merge user heat-maps into one for analysis. This step is performed using MongoDB provided MapReduce pipeline, while putting the processing close to database can increase performance. Figure 13 shows how heat-map can be merged by using MapReduce.
The input is the documents in the database collection "user_heatmaps" with the create time item larger than the last aggregated time, to process only the incremental data. The mapper will map each document to the composite key of video ID and time, the value consisting of heat-map and population. Then the reducer will reduce the value with the same key, that is, adding the values for each corresponding pixel of heat-map and the population. In addition, we finalize the pixel values at the last to be divided by the population, to get the average value, the final result of the merge heat-map will be the average of every user heat-maps.

4.3.2.5 Loading AOI

The AOI loading step joins the AOI heat-map with the merged heat-map, so that they can be compared. Since heat-map data is relatively large, joining is used rather than embedding which can be more space efficient. Such join is performed using MongoDB built-in aggregation pipeline.

After AOI loading, the output of the join-operation will be stored into a temp table "temp_merged_heatmaps_with_teacher_ref", every output document will contain both students’ merged heat-map and teacher’s AOI heat-map.

4.3.2.6 Generating Coverage Report

Every document in the previous step now contains both students’ merged heat-map and teacher’s AOI heat-map. In this step, the total area, student’s area, student’s only area, teacher’s area, teacher’s only area will be calculated.

For every document produced in previous step, the program goes through each pixel in the both heat-maps, counters for each areas are implemented. If the value is below a threshold, this pixel is considered to be covered, we consider this pixel is covered by the gaze. We can obtain student’s area and teacher’s area. Moreover, for student’s only area and teacher’s only area, we increment the counter when the pixel is possible for one heat-map and negative for another heat-map. While the max value of each pixel is 255, the threshold is set to be 5. The output will store in a temp collection.

Finally, the aggregation pipeline groups the heat-map’s areas into reports.
4.4 Client Application

The client application is responsible for interacting with users. Users access system functionalities through client application, they can watch videos and view gaze feedback analysis reports.

4.4.1 User Interface

Users can interact with our system through user interface. For the appearance of user interface, our implementation bases on the theme Material Admin by rushenn. The user interface follows the fundamental layouts and elements of mainstream MOOC platform.

![User Interface of MOOC-Gaze](image)

The functionalities of our user interface are relatively simple. Although two roles are defined in our system, i.e. students and teachers. We do not focus on the user access implementation, the user interface does not separate teachers and students, that’s also convenient in demonstration. The focus is to provide the necessary interface to finish the whole process from gaze collection to viewing analysis result.

4.4.2 Gaze Collection

Gaze collection involves several steps. When user want to watch a video, a calibration step need to finish. When user is watching video, gaze collection is performing in the background. The gaze data need to submit when the collection is finished. The client application should handle these steps necessary for gaze collection. For the implementation, the logics are performed by several classes.
The class diagram shows the prototype of objects we have implemented. The Player object prototype is responsible for controlling the HTML video player, it stores the video information and provides method for getting current time of the video. The Context object prototype is responsible for handling gaze estimation logics. In fact, it includes the reference to JSFeatGaze gaze estimation core inside the gazer attributes.

The GazeCollector object prototype, handles the gaze collection task. While the object of GazeCollector is running in the background, it actively gets the current gaze prediction from context object, and get the current time from the player object. Combined two together, the object manages a gaze series (i.e. a gaze collection). Finally, the gaze feedback will be submitted to server. This implementation can successfully separate the video player logics and the gaze estimation logics.

4.4.3 Resolution Independency

During implementation, we find problem in different screen size. As our system supposes to run on different computers, the screen resolution varies over computers. Due to the design of WebGazer, our gaze estimation core JSFeatGaze, based on WebGazer, will output the x and y values correspond to the position of pixel in the screen. These screen positions, which is dependent to size, are hard to aggregate and analyse. As the inconsistency of screen size, all position collected and stored in our system use video position. In the client application, the position in screen will be converted into position in video.
CHAPTER 5. SYSTEM SCREENSHOTS

Figure 16 Course Overview

Figure 17 Table of Contents
Figure 18 Calibration Screen

Swapping Elements:

Problem: Since \( y = x \) overwrites the value of \( y \), \( x \text{ \&} y \) has no effect. The value stored in \( y \), seven, is lost.

Figure 19 Course Video Player
Figure 20: Heat-map Player

Swapping Elements:

Problem: Since \( y = x \) overwrites the value of \( y \), \( x = y \) has no effect. The value stored in \( y \), seven, is lost.

Figure 21: Table of Content for Viewing Reports
Figure 22 Gaze Coverage Report.
CHAPTER 6. EXPERIMENT

6.1 Experiment 1: Heat-map Analysis

6.1.1 Process and Participants

In the first experiment, two participants are invited to watch a video “Swapping Elements”, with the length of 39 seconds. The instruction of video is in English. However, this experiment focuses on how visual factor affects student’s attention, the sound is muted when playing. The experiment is finished in two laptop computers: The Apple MacBook Pro with Retina Display (13 inch) and the Lenovo T460P (14 inch). They are different in screen resolution, quality of webcam and operating system.

The participants came from computer science major, they are expected to have experience in programming and have the background knowledge for watching this video. Based on the gaze feedback of watching, we can see the students’ attention.

6.1.2 Results and Implication

First, the video is demonstrating the swapping of two elements in animation. Our heat-map shows students are focusing (See Figure 23), the bright yellow color indicates a highly focus is gain. This simple screen is easy to get student’s focus.

![Swapping Elements](image)

*Figure 23 Heat-map for the First Screen in Video*
Second, while the video is demonstrating the swapping in the left, the assign statement is showing in the right. The heat-map focus get diverse (See Figure 24), while focusing on the left side, students are scanning the assign statement on the right side.

![Figure 24 Heat-map for the Second Screen in Video](image)

Third, the video is showing a text description. The heat-map distribution is highly centralized into this region (See Figure 25), students are paying their attention on the explanation text.

![Figure 25 Heat-map for the Third Screen in Video](image)

Forth, the video is demonstrating the solution of swapping in animation. Students are carefully watching the animation, the heat-map distributes in average but stays on the left side (See Figure 26).

![Figure 26](image)
Fifth, the video is showing the assign statement. The heat-map continues to show a centralized focus (See Figure 27).

Sixth, the video is showing a text description. The heat-map shows students are looking at (See Figure 28).
6.2  Experiment 2: Gaze Coverage Analysis

Study already shows the clue that students with better learning performance shows larger area of gaze coverage and less AOI misses. Our reports show two measurement of these two factors: student’s coverage inside AOI, and student’s coverage outside AOI. Based on our reports, students’ learning performance over the time can be measured.

6.2.1  Process and Participants

In the second experiment participants are asked to watch the same videos and with the same settings. However, our participants are divided into following categories:

- Teacher: We record teacher’s gaze data as AOI when he watches the video;
- Focused student: We ask computer science student to focus on watching this video;
- Distracted student: We ask student to watch another video in foreground while recording, in order to simulate distraction, so that the gaze does not follow the content.

6.2.2  Result and Implication

![Figure 29 Report for Focused Student](image)

![Figure 30 Report for Distracted Student](image)
Figure 31 Focused Student: Gaze Coverage Inside AOI

Figure 32 Focused Student: Gaze Coverage Outside AOI

Figure 33 Distracted Student: Gaze Coverage Inside AOI

Figure 34 Distracted Student: Gaze Coverage Outside AOI
For focused student, as we can see in Figure 31, a large coverage inside AOI is shown, on average can achieve more than 78.2%. The focused student also shows a normal coverage outside AOI, it is around 40% on average. For distracted student, as we can see in Figure 33, has less coverage inside AOI, around 68.3%. Although it is higher than what we expected, since the distracted gaze position will still have some level of overlapping with AOI. More obviously, the coverage outside AOI is raised to over 62% on average.

Large coverage inside AOI indicates students have looked at most of content which teacher expected they should. Large coverage outside AOI implies that either students focus on the content other than the teacher expected, or other visual in the video distracts students’ attention. Through these two cases, we found that the coverage inside AOI shows some of difference, but not large difference. For the coverage outside AOI, the difference is more easily to notice and reflects the real attention.
CHAPTER 7. DISCUSSION

The gaze estimation core is developed from WebGazer. Based on the local gaze estimation framework, we employ JSFeat object detector and heuristic cropping method on the face image as features extractor to overcome reliability. It gives relatively static input to prediction model, but it also gives larger region around the eyes as input. Employed experience from appearance-based methods studies, we increase training data size and variability, results in unconstraint head pose and more data points in calibration. Overall, we can achieve equivalent or exceeding accuracy as WebGazer.

As future direction, appearance-based methods show its potential on handling variant conditions, as its development and more public datasets are available in future, adopting these methods into gaze estimation solution can bring the quality to the next level, a comprehensive pre-trained model can apply to our system as well. For local model, there is potential to employ other regression models, such as Support Vector Regression (SVR). However, since our solution bases on explicit calibration of local model, most of the existing machine learning libraries depend on server-side computation (e.g. Node-SVM using NodeJS and C++) thus fail to perform estimation in real-time. On the other hand, rare library (e.g. ML.js) provide client-based solution but almost is not yet ready for use. During our testing, we found lots of errors and bugs. While currently advanced machine learning capability is not ready for client, we leave it as future works.

The gaze feedback analysis base on gaze coverage and heat-map to imply students’ attention. In the experiment, we show the heat-map analysis of course video, also show the comparison of gaze coverage between focused student and distracted student. Through our work, feedbacks and insights are available for MOOC teachers, so that they can discover their strategies and improvement for teaching. While our analysis may reflect just one aspect of students’ learning, this is a developing new field, more studies and theories are needed to be discovered. However, our proposed system as well as the process for collecting, storing and analysing gaze data, could serves a as practice for the MOOC applications, could also be a good base of other future works.
CHAPTER 8. ETHICS AND PROFESSIONAL CONDUCT

Our system protects user privacy. For gaze estimation purpose, people facial image will be captured through webcam. Since facial images are regarded as sensitive data for people identity, our system protects user’s privacy by using local model. Facial images will neither be sent to server nor stored in database. Instead, only the gaze data, which is output by the local model, will be sent to server and store in database. While gaze data contain only the positions on the screen over time, thus is insensitive and user privacy is protected.

Our system allows others to use under GNU GPL license. Our system is developed based on WebGazer [17] and JSFeat [19], which is under GNU GPL license and gives us the right to use. Our system also uses heatmap.js, which is under MIT license. Moreover, the theme of our application is based on Rushenn’s work [21], we have the legal right for using under the regular license granted.
CHAPTER 9. CONCLUSION

We propose the MOOC-Gaze solution, which brings gaze feedback available to the MOOC application. With the gaze estimation method, we make both reliability and accuracy to be acceptable for MOOC use cases, while it can be efficiently run on a web-based setting. Moreover, we design analysis method, insights from gaze feedback are able to be discovered. The system employs modern non-relational database and data analysis techniques, enables the manipulation to gaze data. From gaze estimation to providing analytical insights, our solution serves a practice in the field of MOOC applications.
CHAPTER 10. REFERENCES


