



A

Partial Project Report on

“Proctored Online Examination System using Deep Learning and Computer Vision”

Submitted in partial fulfilment of the requirements

for the Degree of

Bachelor of Engineering

in

Information Technology

under

Savitribai Phule Pune University, Pune

by

# Group :06

## Vinaya Borhude Prathamesh Mohite Rupam Patil Aditya Pawar

Under the guidance of

Mrs. Pragati. P. Mahale

DEPARTMENT OF INFORMATION TECHNOLOGY

AISSMS Institute of Information Technology

Kennedy Road, Near R.T.O., Pune – 411 001,

Maharashtra, INDIA

September 2020

AISSMS IOIT, Pune-01



****

#### 

#### 

**CERTIFICATE**

This is to certify that the dissertation entitled

**” Proctored Online Examination System using Deep Learning and Computer Vision”**

Submitted by

## Vinaya Borhude Prathamesh Mohite Rupam Patil Aditya Pawar

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering in Information Technology at All India Shri Shivaji Memorial Societies’ Institute of Information Technology, Pune under the Savitribai Phule Pune University, Pune. This work is done during year 2019-20, under our guidance.

Mrs. Pragati. P. Mahale Dr. Meenakshi A. Thalor Dr. P.B. Mane

Project Guide H.O.D. Principal

Examiner 1 Examiner 2

Place: Pune Date:26/09/2020

****

**ACKNOWLEDGEMENT**

Without the help of guide we would have never succeeded in completing our Project. Their cooperation, encouragement and help, helped us a lot. First and foremost, we wish to record our sincere gratitude to Management of this college and to Dr P. B. Mane, Principal, AISSMS Institute of Information Technology, Pune for his constant support and encouragement.

Our sincere thanks to Dr. Meenakshi A. Thalor, Head, Department of Information Technology, for her valuable suggestions and guidance throughout the period of this report. We express our sincere gratitude to our internal and external guides, Mrs. Pragati P. Mahale for guiding us in investigations for this project. Our numerous discussions with them were extremely helpful. We hold them in esteem for guidance, encouragement and inspiration received from him.

Our sincere thanks to Prof. Jaideep Patil, Project Coordinator for having supported us with the work related to this project. Last but not the least, we wish to thank our parents for helping in our studies in the college as well as for constantly encouraging us to learn engineering.

Vinaya Borhude

Prathmesh Mohite

Rupam Patil

Aditya Pawar

B.E. Information Technology Engineering

Batch (2020-21)

#### Contents

|  |  |  |
| --- | --- | --- |
| **Sr.no.** | Title | **Pg.no.** |
| 1. | List of Figures/Tables | 5 |
| 2. | Introduction | 7 |
| 3. | Theory | 8 |
| 4. | Literature survey | 11 |
| 5. | Problem Statement | 13 |
| 6. | Problem Design | 14 |
| 7. | Architecture | 23 |
| 8. | Algorithms | 27 |
| 9. | Working Modules | 40 |
| 10. | Plan | 43 |
| 11. | Conclusions | 45 |
| 12. | References | 46 |

**Appendices**

|  |  |  |
| --- | --- | --- |
| **Sr.no.** | **Title** | **Pg.no** |
| 1. | Plagiarism report | 48 |
| 2. | Base papers | 49 |
| 3. | Tools used | 78 |

**Chapter 1**

##### List of Figures

|  |  |  |
| --- | --- | --- |
| **SR. NO.** | **Name of Figure** | **Page No.** |
| 1 | Data Flow Diagram,   1. Level 0 2. Level 1 3. Level 2 | 15  15  16 |
| 2 | UML Diagram,   1. Activity Diagram 1 2. Activity Diagram 2 3. Class Diagram 4. Use Diagram 5. Sequence Diagram 6. State Diagram 7. Deployment Diagram 8. Component Diagram | 17  18  19  19  20  21  21  22 |
| 3 | Agile Software Development | 23 |
| 4 | System Life – Cycle | 24 |
| 5 | Convolution process in CNN | 28 |
| 6 | General CNN architecture | 28 |
| 7 | Max pooling in CNN | 29 |
| 8 | Flattening operation in CNN | 29 |
| 9 | Fully connected layer process | 30 |
| 10 | Primary Position of Gaze | 32 |
| 11 | Face Detection Algorithm Architecture | 33 |
| 12 | Mouth Detection | 35 |
| 13 | Object Detection in Instance Segmentation | 36 |
| 14 | Architecture of Multi – Task Network Separation | 37 |
| 15 | Architecture for Speech to Text Converter | 39 |

**List of Tables**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Name of Table** | **Page No.** |
| 1 | Literature Survey | 11 |
| 2 | Location of Facial Parts | 34 |
| 3 | Inference system rules to classify actions by student | 34 |
| 4 | Accuracy of classifying the validation data using SVM with different kernel functions and PCA dimensions | 40 |
| 5 | Accuracy of classifying the validation data using RBF kernel with different PCA dimensions | 41 |
| 6 | Accuracy of classifying audio samples with different Ls lengths | 42 |
| 7 | Accuracy of classifying audio samples using SVM with different kernel functions | 42 |

##### Chapter 2

##### Introduction

Exams are a critical component of any educational program, and online educational programs are no exception. In any exam, there is a possibility of cheating, and therefore, its detection and prevention are important. Educational credentials must reflect actual learning in order to retain their value to society. A typical testing procedure for online learners is the students come to an on-campus or university-certified testing centre and take an exam under human proctoring. New emerging technologies which allow students to take tests anywhere as long as they have an Internet connection.

However, they still rely on a person “watching” the exam-taking. Exposing the long-established method might prove to be unsuccessful to fully prevent academic malpractice during examinations. E-learning has its vital and integral assessment component using online examination. Submitting exams in E-learning has already been done without a proctor present. As a result, students can easily commit academic malpractice during exams, educational institutions with E-learning depend on an examination process on which students take the exam in a physical controlled environment at the institution under a supervised condition, however, this contradicts the concept of the live E-learning environment.

The ability to efficiently proctor remote online examinations is an important limiting factor to the scalability of this next stage in education. Presently, human proctoring is the most common approach of evaluation, by either requiring the test taker to visit an examination centre, or by monitoring them visually and acoustically during exams via a webcam. How-ever, such methods are labour-intensive and costly. Saving time is one of the perks in having an Online examination system, but it also had limitations on dependency to the quality of Internet service leaving both the proctor and the examiners not being able to use the system. Use of m-learning or other remote education continue to increase due to its ability to reach people who don’t have access to campus. A visual verification for the whole exam session is needed in an online exam, therefore a face verification is needed. A remaining problem in face recognition area is the system robustness.

**Chapter 3**

**Theory**

In this proposed model, we will be using a technique known as knowledge engineering. It looks at the metadata (information about a data object that describes characteristics such as content, quality, and format), structure and processes that are the basis of how a decision is made or the conclusion reached. Some of the existing models such as Paper rater check plagiarism of the given text and even improves it grammatically using Artificial Intelligence and Machine Learning algorithms. This model only improves the quality of the given text by removing grammatical errors and checking its vocabulary. Another example is the E-rater Scoring Engine. This engine is used to grade essays written by students. The essays are graded as a whole without subdividing it. The E-rater Engine makes use of Natural Language Processing to evaluate and obtain a final score of the essay. The proposed question answering model makes use of Natural Language Processing (NLP) based Knowledge Extraction to analyse and interpret each statement of the answer. NLP is making a computer program to understand human language. Knowledge Extraction is creating knowledge from structured text such as database, unstructured text such as documents and images with the help of some already existing knowledge or some pre-determined data. In this model, we will be carrying out knowledge extraction only on structured data. Collaborative learning generally means two or more individuals learn some-thing together. Our model makes use of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) in a collaborative manner. CNN is used for image detection and text classification. Although CNN’s are most suitable for analysing images, some recent developments have shown the successful implementation of text classification using CNN. RNNs are typically used for NLP and speech recognition. While reading the text, RNNs can predict and interpret the future content of data using the already read text. RNNs can be further classified into two types Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). In this model, we will be focusing on Bidirectional LSTM (BiLSTM) which is nothing but LSTM that can analyse the text in two ways, from past to future and from future to past a statement. This, in turn, helps in analysing text better and faster as compared to unidirectional LSTM which only analyses text from past to future. Our proposed model will carry out NLP by collaboratively using CNN and BiLSTM to analyse text.

**Problem Definition:**

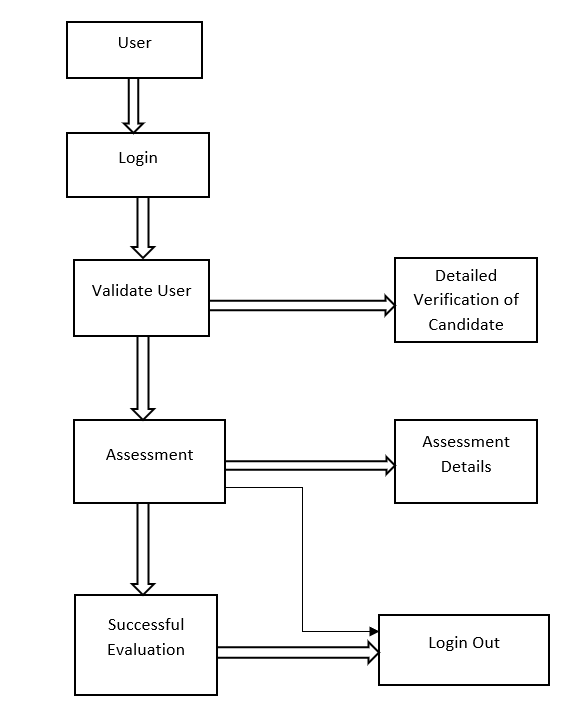
Enormous open online courses offer the potential to significantly expand the reach of today’s educational system, by providing a wider range of educational resources to enrolled students and by making educational resources available to people who cannot access a campus due to location or schedule constraints. Instead of taking courses in a classes on campus, now students can take courses anywhere in the world using a computer, where educators/teachers deliver knowledge via various types of multimedia content. Exams are a critical component of an educational program, and online educational programs are no exception. In any exam, there is a possibility of cheating, and therefore, its detection and prevention are important. A typical testing procedure for online learners is the following: students come to an on-campus or university-certified testing centre and take an exam under human proctoring. The proctors are trained to watch and listen for any unusual behaviours of the test taker(candidates), such as unusual eye movements, or removing oneself from the field of view. They can alert the test taker or even stop the test. In this paper, we introduce a multimedia analytics system to perform automatic and continuous online exam proctoring (OEP). The overall goal of this system is to maintain academic integrity of exams, by providing real-time proctoring for detecting the majority of cheating behaviours of the test taker. To achieve such goals, audio-visual observations about the test takers are required to be able to detect any cheat behaviour. This system monitors such cues in the room where we will monitor candidate using camera and microphones.

**Purpose of System:**

In this project, we focus to develop a multi-tasking analysis system to detect a broad variety of cheating during online examination. System include user verification, text detection, speech detection, active window detection, gaze estimation, face detection, person detection etc. This phase system includes 3 phases:

1. **Preparation Phase:** In this phase, the candidate has to authenticate himself/herself before beginning the exam, by using a password and face recognition. This phase further includes calibration steps to ensure that all sensors are connected and functioning properly. Further, the candidate learns and verbally acknowledges the rules of using the online exam proctoring (OEP) system, such as, he/she has to rotate his/her webcam to 270-360 degree so that we can get the all information about the candidate surroundings. Secondly, no second person is allowed in the same room. There should be light in candidate room so that we clearly perceive his actions, etc
2. **The exam Phase:** The candidate takes the exam, under the continuous monitoring through our OEP system for real time cheating detection. Webcam is used to monitor the user. Using it, we capture images and video used for user verification, gaze estimation, speech detection, text detection, active window detection and phone detection.
3. **Submission Phase:** If user do not found violating any rules and condition then test will be submitted normally after time ends. If user found violating rules, He will be given 3-5number of chances for betterment. If he continue to violating rules again and again then he will be terminate by the system or admin.

Flow Diagram shows the working of Online Proctor System,



**Figure. Flow Diagram of System**

**Chapter 4**

###### Literature Survey

Extraction and mapping of data is done by first identifying the type of data. After the data is identified, analysis and interpretation of the data is carried out with reference to suitable information. The results based on the result show that NLP Unsupervised learning technique is one of the most preferred technique. However, when it comes to large text collection, this technique has certain drawbacks. The system is capable of accepting text as well as speech with the help of a structured query language that is converted from a query by the system. QAVAL [12] is based on answer validation system that selects the most relevant answer for the given question using learning methods. The enhanced lexical and semantic model [16] proposed by Wen-Tau Yih et al, makes use of enhanced lexical semantic models that carries out answer selection using semantic matching with latent word alignment structure. The model’s main sources of errors are inaccurate entity relations, lack of robust question analysis and need of high semantic representation. [7] Data can be extracted through various mediums and one such medium is XML. XML is used in many systems that process data. The main purpose is to experimentally evaluate association rule for mining using XML database. The XML documents are classified; it includes all the information without eliminating any information as in the existing system.

|  |  |  |
| --- | --- | --- |
| **Sr.no.** | **Literature** | **Methodology** |
| 1. | A Deep Learning approach for face detection using YOLO. | YOLO framework for detecting objects. |
| 2. | Automated online exam proctoring. | User verification, speech detection, active window detection, phone detection. |
| 3. | Gaze tracking system using structure sensor and gaze camera. | Gaze tracking, face and eye detection. |
| 4. | Online examination system with cheating prevention using question bank randomization and tab locking. | Royce model for preparation, development, validation, modification and evaluation. |
| 5. | An intelligent system for online proctor monitoring. | Time duration for face disappearance action. |
| 6. | U2Eyes: a binocular dataset for eye tracking and gaze estimation | Binocular images created using UnityEyes |
| 7. | Heuristic based approach for online exam proctoring. | Face detection using inference of activities by user |
| 8. | Face detection and recognition system using digital image processing. | Process of Face Detection System. |
| 9. | Identity aware face super resolution for low resolution face recognition. | Face super resolution network architecture and identity aware loss. |
| 10. | Google API Speech to Text | The conversion of speech to text using Google API |

**Table: Literature Survey**

**Chapter 5**

**Problem Statement**

The title of this project is “**Continuous user verification for online exam proctoring using Machine Learning and Deep Learning**”. It is an online proctoring system which helps organisations, test- takers and various others professional to assess individuals for any specific reason without actual presence of a physical proctor. It can also be used for distance learning and assessment as it is not required for the candidate to be physically present at the assessment centre.

In today’s modern world time is everything. No one can afford any heavy expense of time for a particular work, such as candidate assessment which can be done online. For, this purpose the plan of the project is to develop a system which will continuously proctor the candidate undergoing the assessment and prevent any malpractice that the user may perform and maintain the dignity of candidate assessment.

The user can only undergo the online proctoring assessment on a desktop or a laptop. In this process the user will be prompted to allow mic and front cam to be active throughout the session and the system will keep an eagle’s eye on the candidate. The candidate if performs any malpractice will be warned about the followed misconduct, and if the warning prompts a limited number of times, then user will automatically be logged out of the assessment and won’t be accepted for any further assessment.

**Chapter 6**

**PROJECT DESIGN**

**6.1 Security Requirements:**

To access a new level authentication and authorization by passwords.

* **Perimeter Security**: Network security, firewalls, and, ultimately, authentication to confirm user identities is guided.
* **Data Security**: Data is protected from unauthorized user by providing user login and password credentials.
* **Access Security**: Defining what authenticated users and applications can do with the data in the cluster through providing access to staff via there login and password credentials.

**6.2 Technologies Used**

* Python IDE.
* OpenCV.
* Flask.
* MySQL.

**6.3 Requirements**

* Internet Explorer 9.0 and higher
* Google Chrome latest stable release
* Mozilla Firefox (All Versions)

**6.4 Performance Requirements**

• Accuracy.

• Processing Time

• Memory Usage

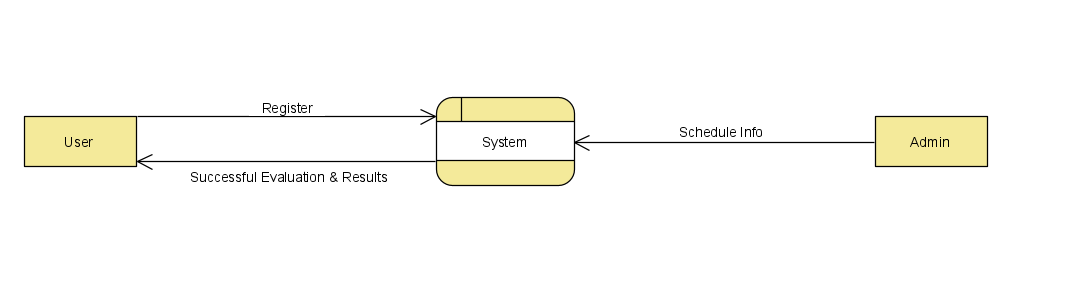
**6.5 Security Requirements**

Authentication and authorization are done strongly, which has data integrity and confidentiality.

**6.2 Data Flow Diagrams**

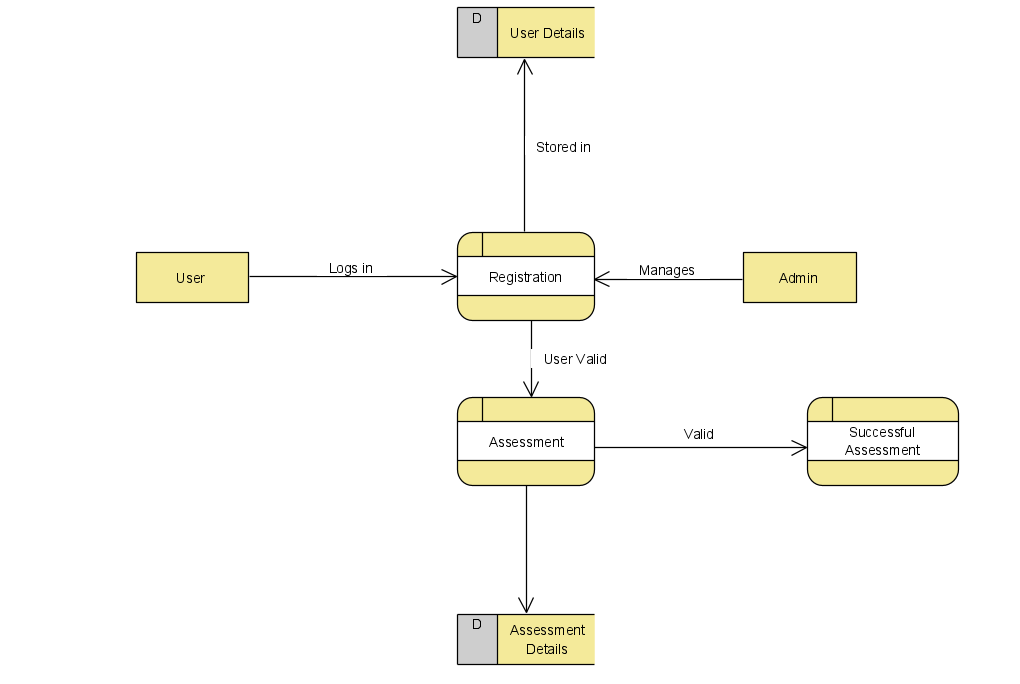
**6.2.1 Level 0:**

It is also known as context diagram. It’s designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as single bubble with input and output data indicated by incoming-outgoing arrows.

****

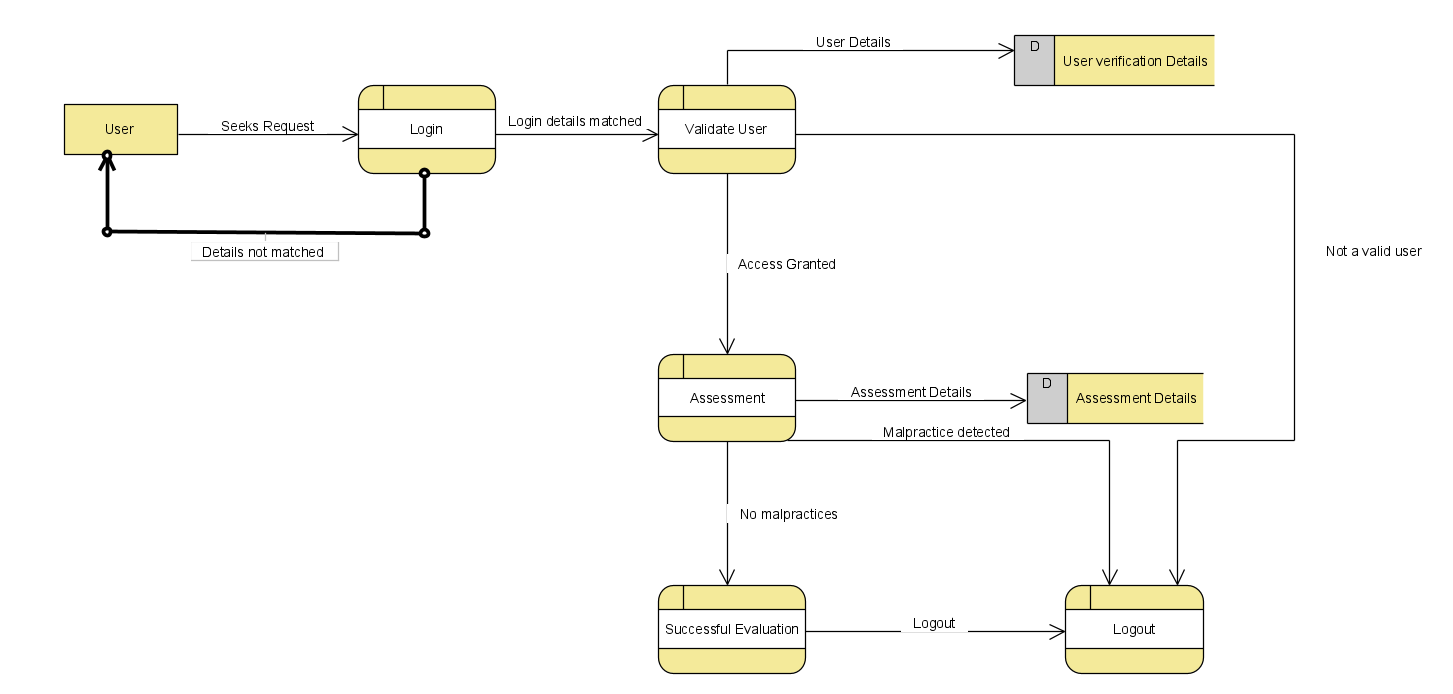
**Figure. Level 0**

**6.2.2 Level 1:**

Context diagram is decomposed into multiple bubbles/processes.in this level we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into subprocesses. ****

**Figure. Level 1**

**6.2.2 Level 2:**

Level-2 DFD goes one step deeper into parts of level-1 DFD. It can be used to plan or record the specific/necessary detail about the system’s functioning. 

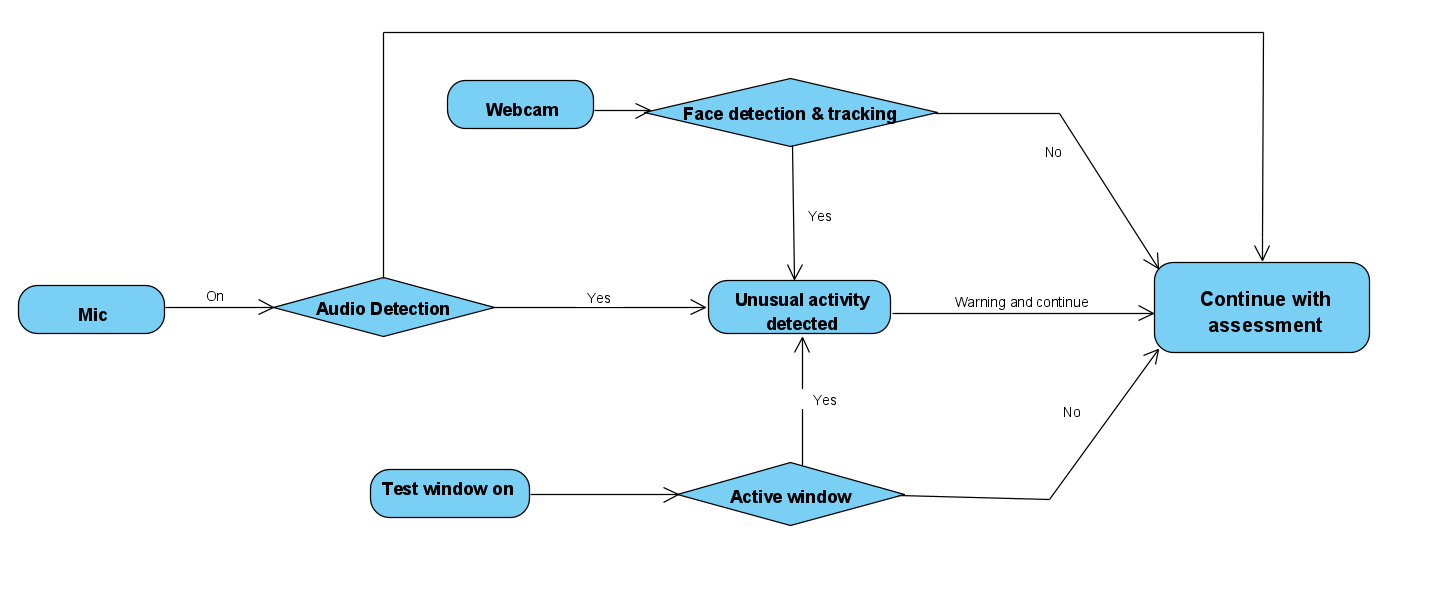
**Figure. Level 2**

**6.3 UML Diagrams**

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

**6.3.1 Activity diagram 1:**

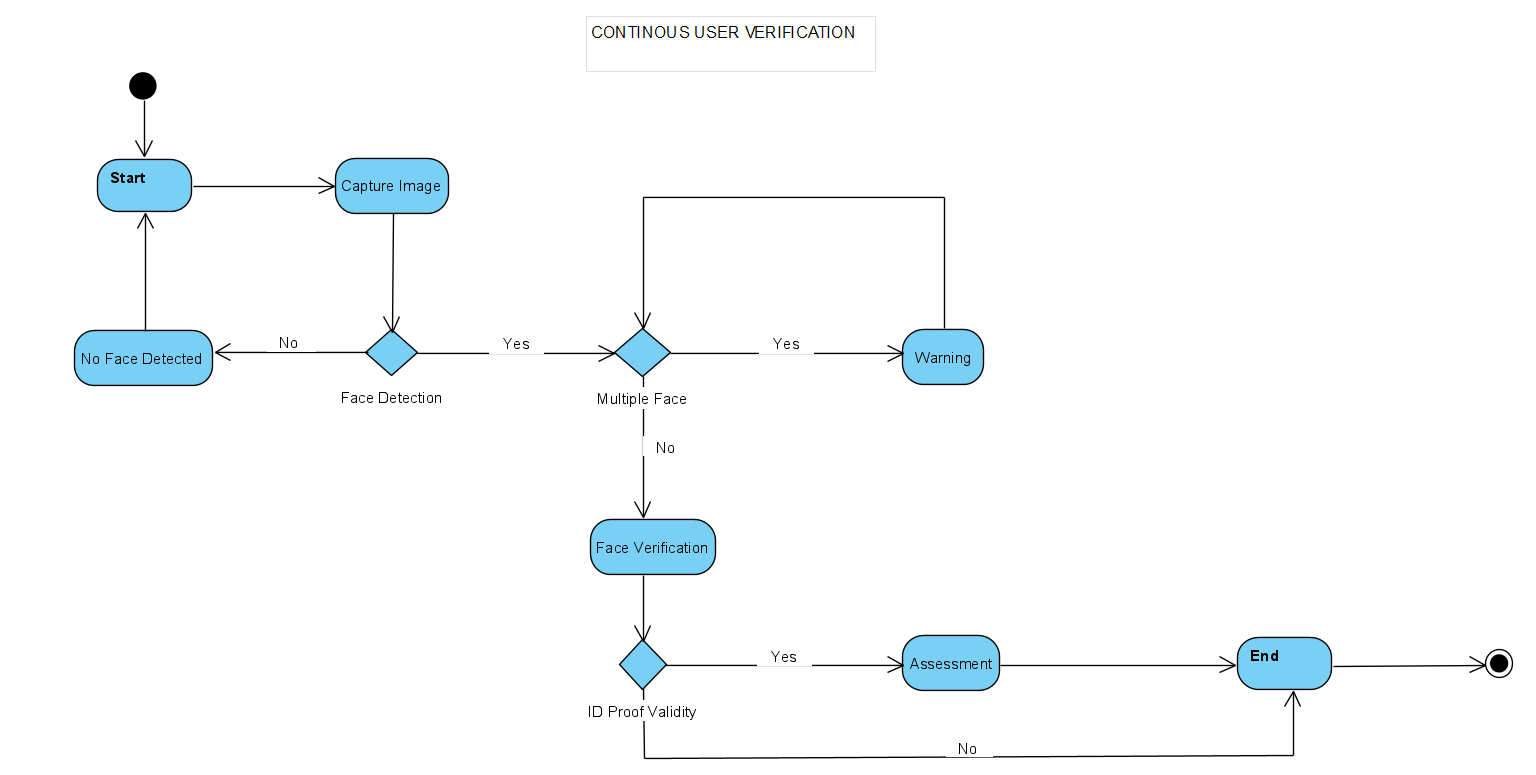
This diagram depicts the entire system in a modelled way such that they are in an outer look.

****

**Figure. Activity Diagram 1**

**6.3.2 Activity diagram 2:**

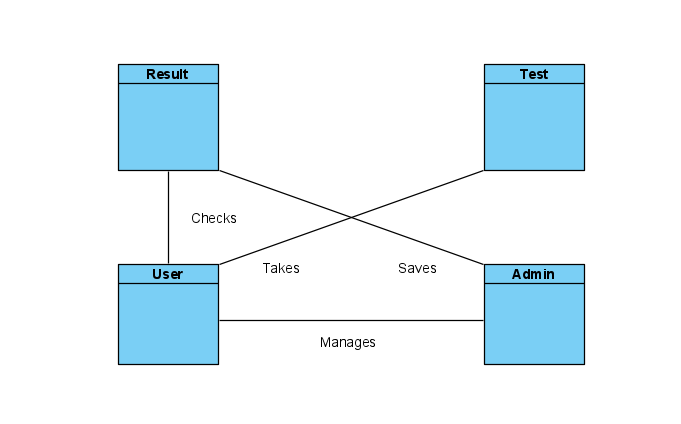
It models the entire process of user verification throughout the system.

****

**Figure. Activity Diagram 2**

**6.3.3 Class Diagram:**

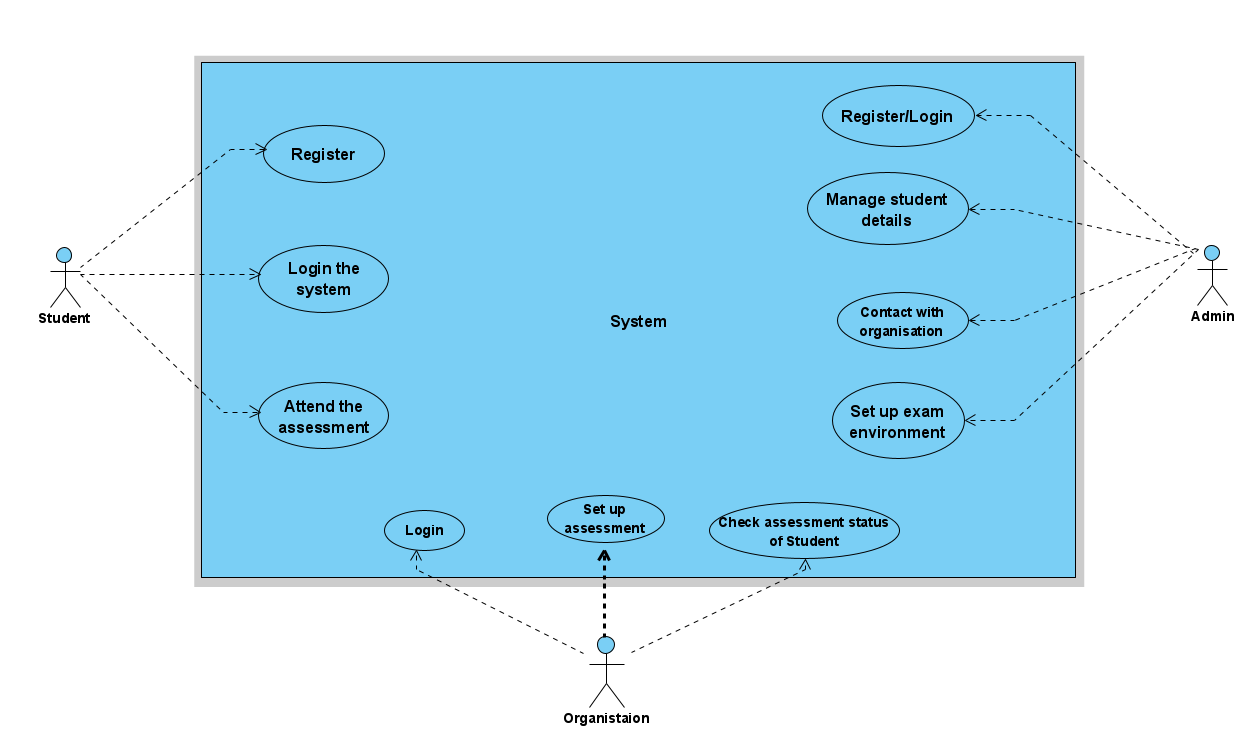
It is the main building block of object-oriented modelling. It is used for detailed conceptual modelling and building of the system. Almost all of the UML models are developed by considering this model as a base model. A class diagram can be used to display logical classes, which are typically the kinds of things. It can also be used to show implementation classes, which are the things the programmers typically deal with.



**Figure. Class Diagram**

**6.3.4 Use Case:**

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. It provides the simplified and graphical representation of what the system must actually do.

****

**Figure. Use Case**

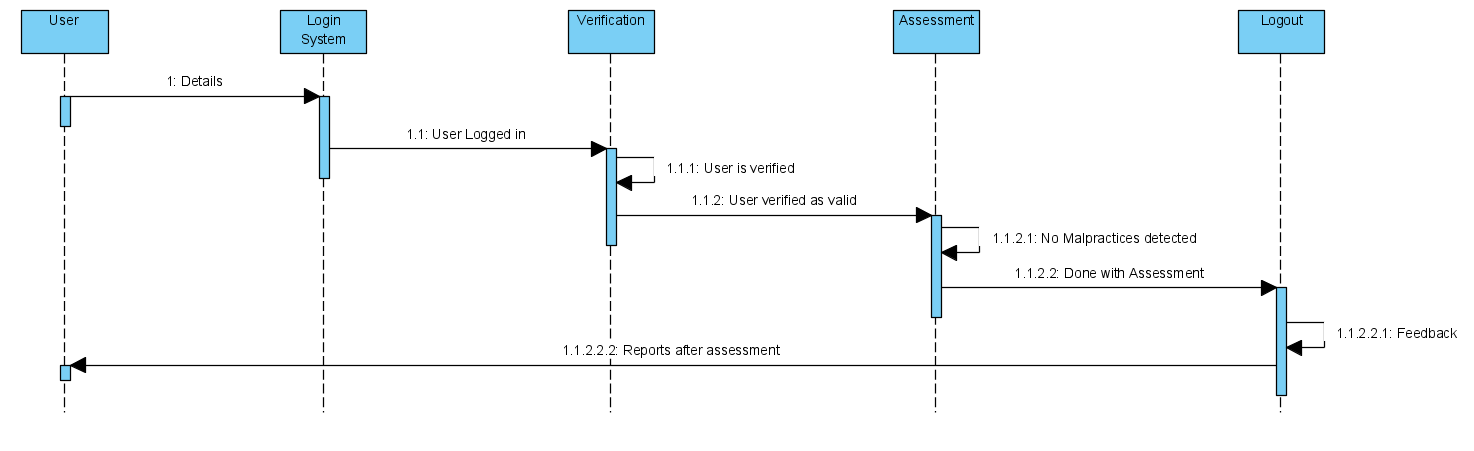
**6.3.5 Sequence Diagram:**

Sequence diagram show a detailed flow for a specific use case or even just a part of it. It explains all the details of the objects in the sequence in which they are executed in a detailed level.

It has two dimensions

**1. Vertical Dimension:** It shows the sequence of messages in the order in which they occur.

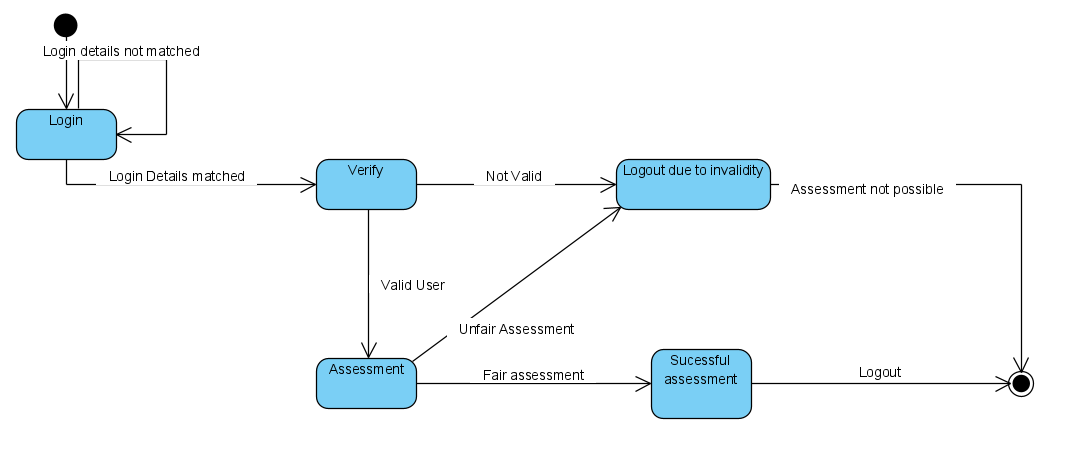
**2. Horizontal Dimension:** It shows the object instances to which messages are sent.

****

**Figure. Sequence Diagram**

**6.3.6 State Diagram:**

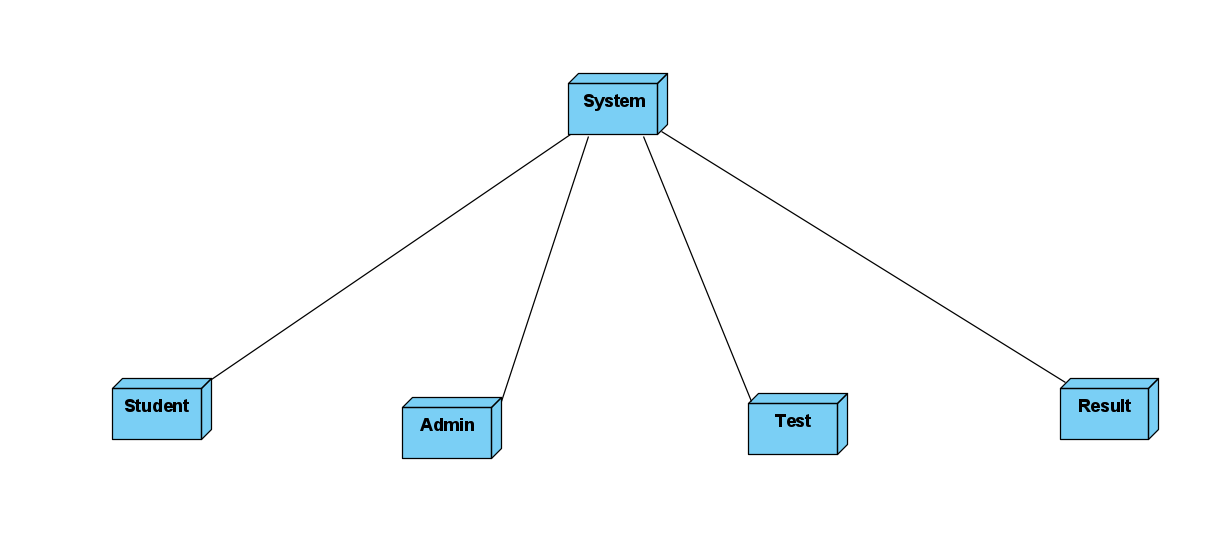
A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It is a behavioural diagram and it represents the behaviour using finite state transitions**.**

****

**Figure. State Diagram**

**6.3.7 Deployment Diagram:**

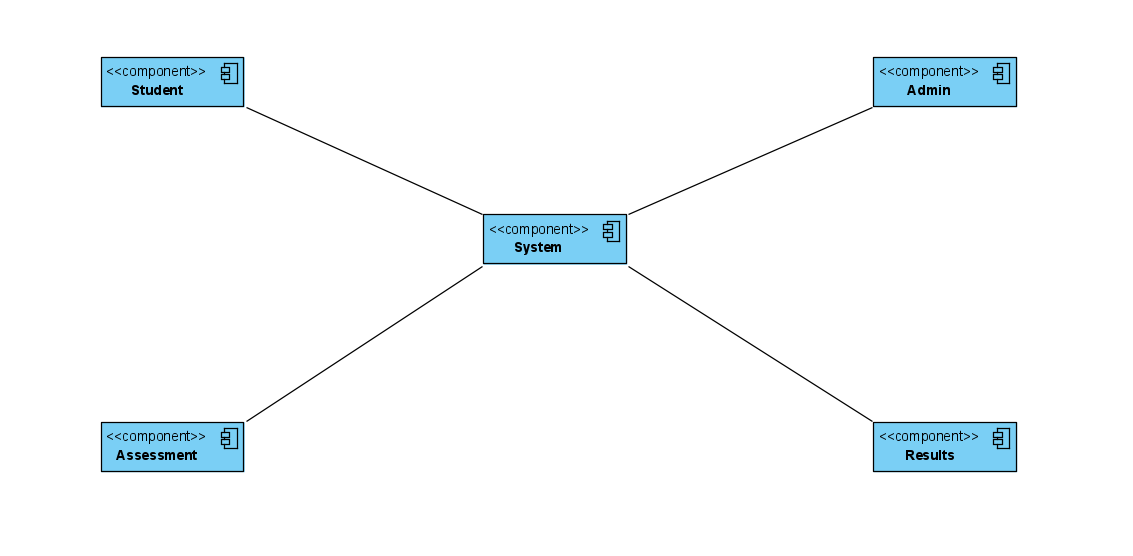
A deployment diagram models the physical deployment of artifacts on nodes. To describe a system a deployment diagram would show what hardware components exist, what software components run on each node, and how the different pieces are connected.



**Figure. Deployment Diagram**

**6.3.8 Component Diagram:**

A component diagram allows verification that a system's required functionality is acceptable. These diagrams are also used as a communication tool between the developer and stakeholders of the system. The diagrams formalize a roadmap for the implementation, allowing for better decision-making about task assignment or needed skill improvement.



**Figure. Component Diagram**

**Chapter 7**

Methodologies

**The seven phases for Software Development are:**

1.Requirement Analysis

2.Feasiblity Study

3.Design

4.Coding

5.Testing

6.Installation and Deployment

7.Maintenance

****

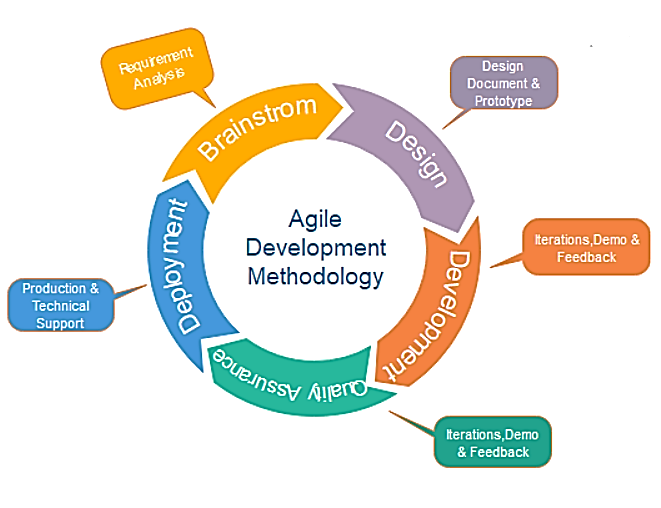
Figure. Agile Software Development

***What is Agile?***

  Agile is the ability to create and respond to change. It is a way of dealing with, and ultimately succeeding in, an uncertain and turbulent environment. It approaches development requirements and solutions through the collaborative effort of self-organizing and cross-functional teams and their customers/end users. It advocates adaptive planning, evolutionary development, early delivery and continual improvement and it encourages flexible responses to change.

***Why agile?***

It gives more capabilities than just Extreme Programming and Feature Driven Development. It widely uses the set of frameworks and practice based on the values in Manifesto for Agile Software development. By breaking down the project into manageable units, the project team can focus on high-quality development, testing, and collaboration. Agile helps project teams deal with many of the most common project pitfalls.

****

**Figure. System Life-Cycle**

We use the agile development methodology to minimize risk (such as bugs, cost overruns, and changing requirements) when adding new functionality. In all agile methods, teams develop the software in iterations that contain mini-increments of the new functionality. There are many different forms of the agile development method, including scrum, crystal, extreme programming (XP), and feature-driven development (FDD).

**Pros:** The primary benefit of agile software development is that it allows software to be released in iterations. Iterative releases improve efficiency by allowing teams to find and fix defects and align expectation early on. They also allow users to realize software benefits earlier, with frequent incremental improvements.

**Cons:** Agile development methods rely on real-time communication, so new users often lack the documentation they need to get up to speed. They require a huge time commitment from users and are labour intensive because developers must fully complete each feature within each iteration for user approval.

Agile development methods are similar to rapid application development (see below) and can be inefficient in large organizations. Programmers, managers, and organizations accustomed to the waterfall method (see below) may have difficulty adjusting to agile SDLC, so a hybrid approach often works well for them.

How did we use agile:

Agile development is based on the ability to create working iterations , test them and find any faults , then create another iteration with those faults fixed , test them , repeating this process until a final polished version of the product is ready .

The first half of the development phase had 3 main phases as follows :

The initiation phase

The initiation phase is the first phase of the entire project management life cycle. The goal of this phase is to define the project, develop a business case for it, and get it approved.

The planning phase

The planning phase is critical to creating a project roadmap the entire team can follow. This is where all of the details are outlined and goals are defined in order to meet the requirements laid out by the development team.

The execution phase

This stage is where the meat of the project happens. Deliverables are built to make sure the project is meeting requirements. This is where most of the teams efforts are pulled into the project.

For the initial half of the development phase, following modules were implemented:

Based on this agile methodology, we created iterations of every working module as follows:

**1. Login Module (Student and Organization Login):**

Here, user enters their login details, which are checked for their occurrence in the database, and upon successful verification, allows the user to access their account.

**2. Registration Module:**

Based on the user type (student/organization), user enters required info, which is then stored in the database.

**3. Examination Module:**

This is the most important and complicated module, which required multiple iterations for a properly working module. In this module, students appear for examination, making it the extremely important to find and fix any possible exploits.

However, face detection and feature tracking are not implemented in any iterations, as they are scheduled for the second half of the development process.

**Chapter 8**

**Algorithms**

**Continuous User Verification**

Continuous verification becomes important in an online exam, as users must be continuously verified during the exam session. A number of characteristics of existing biometrics can be used for various applications. Each biometrics have its own advantages and disadvantages, and the selection of biometrics to be used will depend on the application being developed. Biometric recognition involves large computation for its performance so does in face recognition when additional image processing, such as video analysis, pose normalization.

equalization, etc., which can slow down the system are involved. Another limitation exists in some smartphones is its memory limitation so we cannot use algorithms that utilize a large amount of memory to run. The proposed method is expected to satisfy the need for accurate, low computation cost, inexpensive, convenient online exam proctoring for m-learning performed on smartphones. Therefore the must have the ability to fulfill the requirement as follows:

**Training Set Method:**

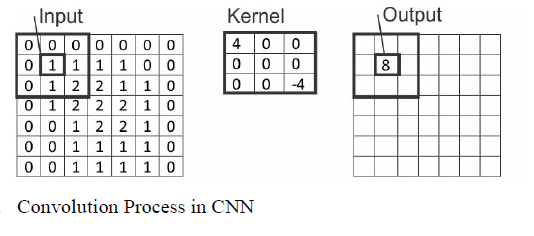
The training process of a face verification needs a significant amount of face samples and non-face samples. The large size of the training set makes it hard to train all samples at the beginning. Therefore in the proposed method, incremental training will be used for face dataset. The dataset itself is

obtained from every online lecture session followed by the user using his mobile device, while the training of the dataset will be done in the server.

To train the image dataset a Convolutional Neural Networks (CNN) machine learning will be performed. CNN is one type of artificial neural network commonly used in image data. Common CNN is different from artificial neural networks in general. CNN consists of neurons that have weight, bias and

displacement functions. CNN describes three main architectures, namely local receptive fields, sharing weights consisting of filters, and spatial subsampling consisting of unification. CNN uses a filter also known as the kernel, to

identify what features are in the image area.

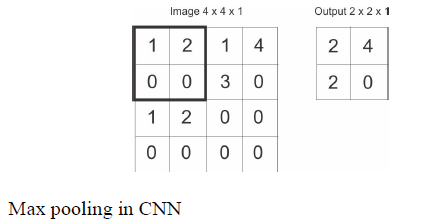


Filters are valued matrices, called weights, which are drilled for the conversion of certain features. To provide a value that indicates a particular feature, the filter performs a convolution operation described in the output of the convolution process is summed with the term bias and passes through the non-linear activation function. One of the activation functions commonly used is the Rectified Linear Unit (ReLU) activation function. CNN has several layers that serve to filter what has been determined during the training process, namely convolutional layer, pooling layer, and fully connected layer.

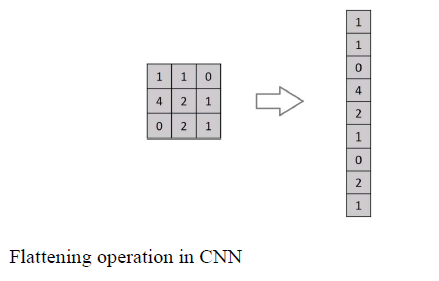


Down sampling needs to be done to accelerate the training process along with reducing the memory consumed by the network. There are several ways that can be done, one of the most common ways is max pooling . In max pooling, a

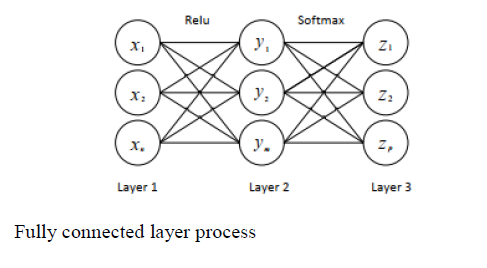
matrix window traverses on the image based on the set of steps. The maximum value in every step window is selected then combined into an output matrix.



After convolution and pooling layer operations are finished, the image representation is passed into a fully-connected layer after it is flattened into a vector of feature to predict the probability of output. Fig. 4 describes flattening operations.

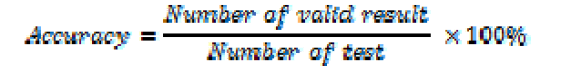


The softmax activation function will process the output of the dense layer by mapping all the final output layer into a vector with one element. Fig. 5 shows the process carried out in a fully-connected layer. Layer 1 will feed-forwarding to layer 2 using the RelU activation function. In layer 2, a classification will be done using softmax.

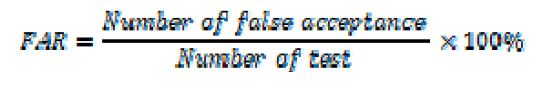


**Evaluation Method:**

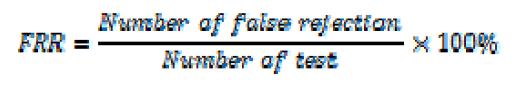
The evaluation is executed both in the server and using the user’s mobile device when the online exam took place. The parameters to be evaluated as mentioned earlier are the accuracy rate, FRR, and FAR. The accuracy rate is done by dividing a valid sample with the number of trials multiplied by 100%



False acceptance rate (FAR) is an error in recognizing/accepting the identity of the input image as a valid user whereas it was an invalid.



False rejection rate (FRR) is an error in rejecting an input image, where an input image that should be recognizable turns out to be unrecognized/rejected.



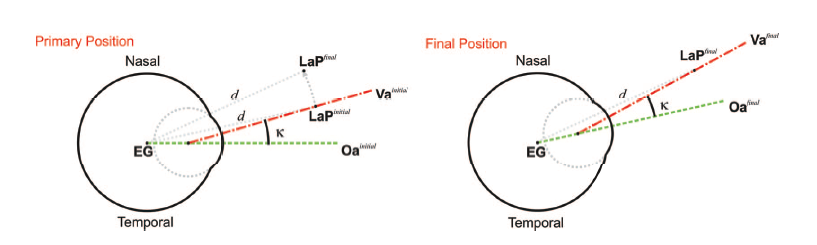
The system will have five vision-based capabilities which are combined using multithreading so that they can work together:

**1.Eye detection and Gaze tracking:**

Gaze tracking algorithm first detects the face of a user on the RGB images captured by the webcam based on the Viola- Jones algorithm. But because the user is far away from the webcam, the face image is understandably low resolution. Therefore, the proposed gaze tracking algorithm detects eyes on the face candidate images, instead of repeatedly detecting a face in reducing the expected size of a face. In addition, it improves the eye detection performance by expanding the size of the face image with interpolation. After the face and eye detection, it tracks them using Kalman filter to reduce the processing time. Data collection by either a remote, head-mounted or VR based ‘eye tracker’ connected to a computer. Eye trackers generally include two common components: a light source and a camera. The light source (usually infrared) is directed toward the eye. We shall aim to track the eyeballs of the test-taker and report if he is looking to the left, right, or up which he might do to have a glance at a notebook or signal to someone.

This can be done using D-lib’s facial key-point detector and OpenCV for further image processing. Among the obstacles that eye tracking community encounters when facing the challenge of low-resolution gaze estimation, the lack of large-scale labelled datasets to be used for these purposes is remarkable. Ideally, datasets including images of the eye/face area are required where not only face but also eye area landmarks (eyelids, iris, pupil) are included. Moreover, images should be annotated with gaze information and, preferably, head pose should be also labelled. Many efforts have been made by researchers in order to generate large datasets containing the corresponding labels. However, although deep learning techniques proved to be successful in most areas of research, the accuracies obtained using these datasets in terms of gaze estimation are insufficient. One of the hypothesis is that the models do not learn to generalize because datasets employed for training purposes lack of enough variability. In order to enlarge these datasets size and trying to avoid the burdensome manual labelling option other possibilities have been proposed, such as image augmentation techniques or synthesizing images. Gaze tracking algorithm detects glints which are corneal reflections generated by the two NIR illuminators and utilized as a reference point for gaze position calculation.

It first detects bright regions in the IR image using adaptive threshold method. And then, it differentiates them between square regions and long rectangle regions where two glints are located closely. Especially in the case of square regions, it finds another square region connected from side to side, and then make the two regions into a long rectangle region. Next, it finds the centre of each long rectangle region using Gaussian blurring. The long rectangle region whose centre is closest to the pupil centre is decided as real glints. Gaze tracking algorithm also detects pupil centre. It first specifies pupil candidate regions of the foregoing bright regions using morphology dilation and pupil extraction filter that has a square shape and compares all the border pixels are brighter than the centre. And it verifies all the pupil candidate regions by checking the existence of the glints. Finally, it extracts the pupil centre using rough circle estimation based on sobel filter and weighted ellipse fitting which controls contour points before applying ellipse fitting especially for excluding glints eyelid.



**Figure. Primary Position of Gaze**

In the upper part of the figure the primary position is shown together with the LaP in the final position. An imaginary axis is calculated in both, initial and final positions connecting EG to LaP*initial* and LaP*final* respectively. After estimating the rotation between the imaginary axis it is applied to the whole eyeball. In the lower part, both optical and visual axes have been rotated accordingly to the final position. Now, the visual axis points to the pursued LaP*final*.

**2.Facial Landmark Detection**

One of the sub divided image frame makes one class i.e. the one consisting the faces in the image, which marks the first step towards the process of face detection. It is inconvenient because in spite of the congruity exist among faces but several factors like age, skin colour and facial expression can vary considerably. Then this problem is furthermore intricate by the arrival of factors like environment factors affecting light, risk of imitation and also

probability of limited obstruction in image. The face detection system that can easily recognize any face from a given image that too under any circumstance

with any kind of lighting environment is thus considered as the finest face detection system. The function of the face detection system can be further

bifurcated into two phases. Phase one consists of classification, in which the system based on the input that was in the form of some random images and if

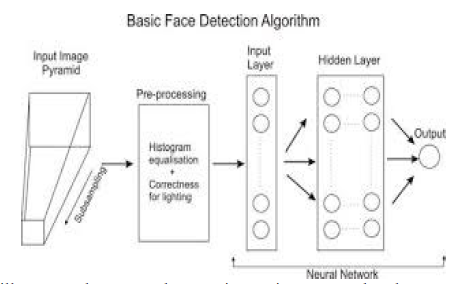
the face is present in the image the output comes in the form of yes or no. Face localization is the second phase in which for a given input image it shows a bounding box which comprise the dimensions of exact location of the face in the image.

The process of face detection system is sub-divided as follows:

1) **Pre-Processing:** Before feeding any image to the network it is processed properly to lower down the variability. Frontal faces that are comprised in the front view of the image is thus obtained by cropping the images that contain the human faces. On completion of the above step, standard algorithms are used to correct the lighting of the cropped images.

2) **Classification:** To categorize any image as faces or non faces, neural networks are implemented by training on these examples. For the process of

classification, we have combined the MATLAB NN toolbox along with the basic implementation of the neural networks.



**Figure. Face Detection Algorithm Architecture**

The facial landmark detector which is pre-trained inside the d-lib library of python for detecting landmarks, is used to estimate the location of 68 points or (x, y) coordinates which map to the facial structures. These indexes of 68 coordinates or points can be easily visualized on the image.

The Locations of the Facial Parts are shown in table:

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Facial Part** | **(x Co-ordinate)** | **(y Co-ordinate)** |
| 1 | Left Eye | 42 | 48 |
| 2 | Mouth | 49 | 67 |
| 3 | Left Eyebrow | 22 | 26 |
| 4 | Nose | 27 | 34 |
| 5 | Right Eye | 17 | 21 |
| 6 | Right Eyebrow | 36 | 41 |
| 7 | Jaw | 0 | 16 |

**Table. Location of Facial Parts**

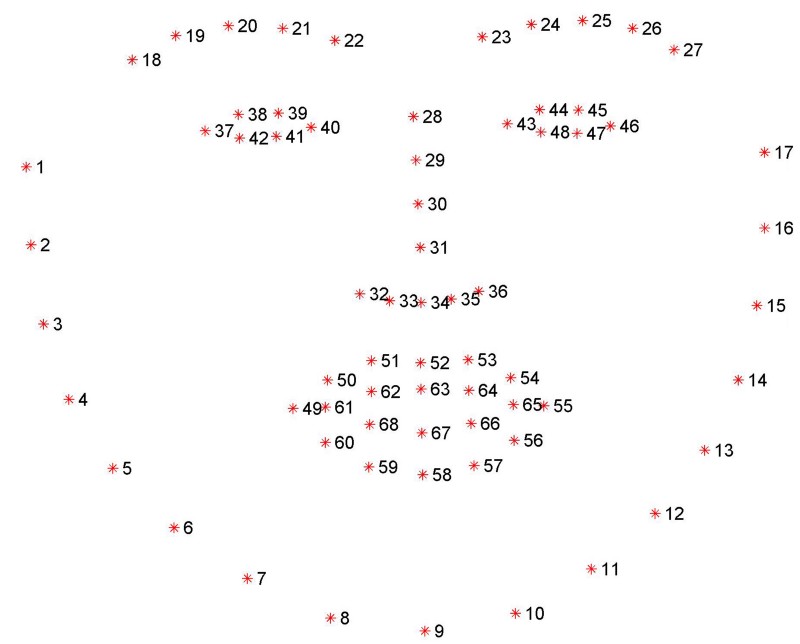
The first part of proctoring is face detection. To evaluate face detection in our system we first use face video dataset to train our system. After detecting face the next step is to track face and ensure continuous presence of student in the exam. Since there are a lot of face movements of face during online exam we use dataset for evaluating face features. The following table gives set of rules that can be used for evaluating face tracking.

|  |  |  |
| --- | --- | --- |
| Sr. No. | Rules | Decision |
| 1 | Detected any face missing from frame at any point of duration | Malpractice |
| 2 | Face moving far away from webcam more than 2 times | Warning |
| 3 | Face moving far away from webcam more than 4 times | Malpractice |
| 4 | Multiple face detected | Malpractice |
| 5 | Sound more than mean amplitude for more than 1 time | Warning |
| 6 | Sound more than mean amplitude for more than 3 times | Malpractice |
| 7 | Any window active other than browser | Malpractice |

**Table: Inference system rules to classify actions by student**

**3.Mouth Detection**

* This is very similar to eye detection. D lib’s facial key-points are again used for this task and the test-taker is required to sit straight (as he would in the test) and the distance between the lips key-points (5 outer pairs and 3 inner pairs) is noted for 100 frames and averaged.
* If the user opens his/her mouth the distances between the points increases and if the increase in distance is more than a certain value for at least three outer pairs and two inner pairs then infringement is reported.
* The mouth can be accessed through points [49, 68].



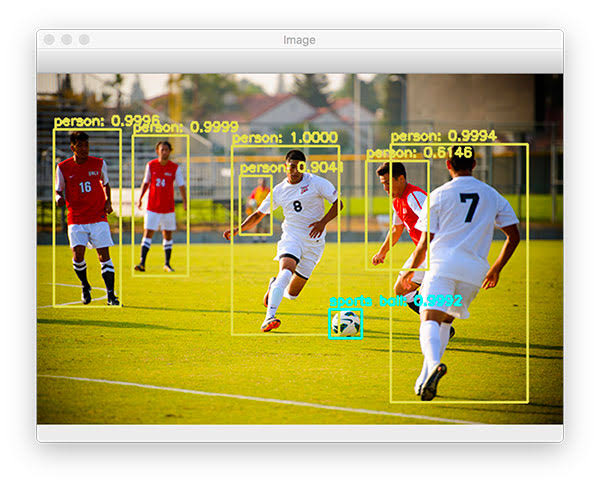
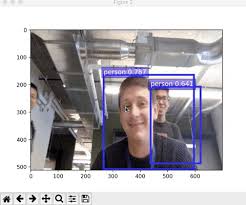
**Figure: Mouth Detection**

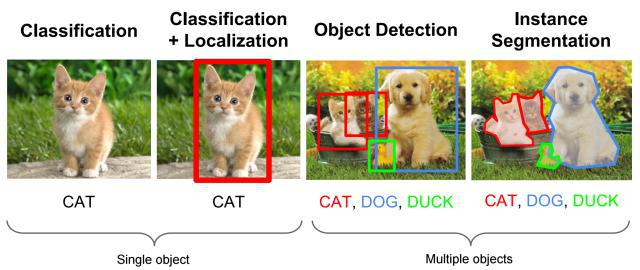
**4.Person Counting and Mobile Phone Detection**

In this, we are focusing on the problem of detecting each instance of a specific category of person. A new technique use for person detection is proposed based on a deep counting model. The feature extractor in YOLOv3 of the deep counting model is extended with additional layers for segmenting specific instances and also focus on the persons in the scene. The segmentation layers help to get a more accurate estimation of the foreground with persons. The instance segmentation is able to estimate separate instances of persons.

YOLOv3 (You Only Look Once, version 3) will perform multiple image analysis tasks as person counting, person segmentation and instance segmentation. A deep counting model have features extensions, use to obtain a comprising small number of layers to multi task network which helps to achieves segmentation combination with tasks. By using frame-work we obtain the output for segmentation maps and semantic boundaries to use in post processing steps. YOLO results in the feature extractor that useful for analyse applications as shown in figure. Figure shows the feature extractor of YOLO Algorithm trained for counting persons, able to focus on persons in the different

scenarios.



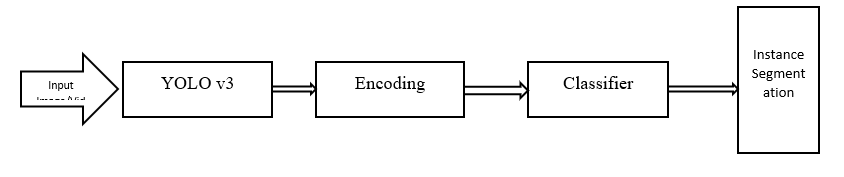


**Fig. Object Detection in Instance Segmentation**

The architecture of a comprises of a common feature extractor that followed by three layers that are, 1) A specific layer for object counting 2) A second layer for segmentation, and 3) A third for instance separation.

The feature extractor of the deep counting model is able to focus on the foreground region. We extending it to few layers to achieve the segmentation and instance separation instead of a full-fledged decoder comprising of a series of de-convolution and un-pooling layers. This helps to reduce the number of computations during inference. The probabilistic segmentation map drives the head for instance segmentation. The instance separator function is to create an embedding in a feature space such as the pixel representation corresponding to the same instances that are close together and others are separated. Due to the number of instances in each frame vary and no clear discriminant, Simple classifier or cross entropy loss is not possible to achieve instance segmentation directly. Also, the ordering of the instances is not important. The task of instance segmentation is broken down into background removal, foreground localization and distance separation that followed by post-processing steps for instance segmentation.

Instead of using separate networks for each of these sub-tasks, using a combined network that comprises of a common feature extractor is beneficial. The different components of the network are synergistic and assist each other during the training resulting in quicker convergence.



**Figure. Architecture of Multi-Task Network Separation**

During inference, the trained model can describe the prediction of the count of instances and the instance separated output embedded in an n-dimensional space. To obtain the instance segmentation, clustering techniques like mean shift algorithm or K-means can be used in which the predicted count can be used to initialize the number of clusters. A deep network with few layers that has been trained for obtaining instance segmentation from the embedded space can be used. The predicted object count will be use to select the appropriate network.

**5.Audio to Text**

The idea is to record audio from the microphone and convert it to text using Google’s speech recognition API. The API needs a continuous voice from the microphone which is not plausible so the audio is recorded in chunks such there is no compulsory space requirement in using this method (a ten-second wave file had a size of 1.5 Mb so a three-hour exam should have roughly 1.6 Gb). A different thread is used to call the API so that a continuous recording can without interruptions, and the API processes the last one stored, appends its data to a text file, and then deletes it to save space.

Speech to text has three main methods to perform speech recognition:

**1.Synchronous recognition** send audio data to the speech to text API, performs recognition on the data and results after all audio has been processed. Synchronous recognition requests are limited to audio data of 1 minute or less duration.

**2.Asynchronous recognition** sends audio data to speech to text API and initiates a long running operation. Using this operation, you can periodically poll for recognition results. Use asynchronous results for audio data of any duration up to 480 minutes.

**3.Streaming Recognition** performs recognition on audio data provided within a gPRC bi-directional stream. Streaming requests are designed for real time recognition purposes, such as capturing live audio from a microphone. Streaming recognition provides interim results while audio is being captured, allowing result to appear. Requests contain configuration parameters as well as audio data.

**SPEECH TO TEXT API RECOGNITION:**

A speech to text API recognition request is the simplest method for performing recognition on speech to audio data. Speech to text can process up to 1 minute of speech data audio sent in a synchronous system. After speech-to-text processes and recognizes all of the audio, it returns a response.

A synchronous request is blocking, meaning that Speech to Text must return a response before processing the next request. Speech to Text typically processes audio faster than real-time, processing 30 seconds of audio in 15 seconds on average. In cases of poor audio quality, your recognition request can take significantly longer.

Synchronous speech recognition requests

A sample speech to text API request:

{

“config”: {

“encoding” : ”LINEAR16”,

“sampleRateHertz”:16000,

“languageCode”: ”en-US”,

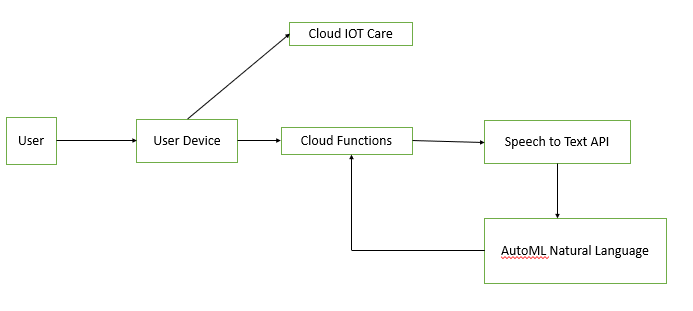
},

“audio”: {

“uri” : ”gs://bucket-name/path/path-to-audio-file”

}

}



**Figure. Architecture for Speech to Text Converter**

**Chapter 9**

**Experimental Results**

We design experiments to answer the following questions: 1) How well can the system detect cheating? 2) How do different feature sets affect the performance? 3) What is the detectability of each cheat type? 4) Is there any correlation between the six components of the OEP system? 5) What is the system efficiency at a component and system level? We now discuss different aspects of our experiments. We start by explaining the evaluation procedure.

Then we analyse the individual performance for a couple of basic components of our OEP system. After that, we test the performance of the entire OEP system. Finally, we describe the OEP system efficiency.

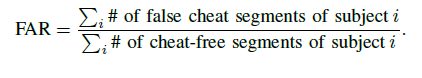
**A. Performance Evaluation**

We define two metrics to evaluate the OEP system, a segment-based metric and an instance-based metric. The segment-based metric evaluates the estimated classifier decisions at the segment level, which is the most straightforward measurement of the classification accuracy. A cheating instance is defined for the entire duration of one continuous cheating behaviour, regardless of how long it is. The instance-based metric evaluates the detection accuracy based on the unit of cheating instance. Therefore, it is the “perceived” system accuracy of the user, and can answer questions such as “if a test taker cheats 10 times, how many times can OEP detect?” Both segment- and instance-based metrics are represented by True Detection Rate (TDR) and False Alarm Rate (FAR), but computed in different ways.

**a) Segment-based metric:** For segment-based metric, TDR is calculated by:



where i denotes the test subject ID. Since it is also important to not claim that a test taker is cheating when he/she is not, we compute FAR by:



|  |  |  |  |
| --- | --- | --- | --- |
| **Kernel/Dim** | **50** | **100** | **200** |
| Linear | 86.85 | 85.63 | 88.09 |
| Quadratic polynomial | 73.55 | 73.61 | 74.53 |
| Cubic Polynomial | 78.61 | 81.91 | 74.50 |
| Radial basis | 93.38 | 93.43 | 94.25 |
| Sigmoid | 81.51 | 83.94 | 82.74 |

**TABLE : Accuracy of classifying the validation data using SVM**

**with different kernel functions and PCA dimensions.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Gamma/Dim** | **50** | **100** | **200** |
| 0.1 | 83.94 | 85.21 | 84.73 |
| 1 | 92.63 | 93.82 | 93.02 |
| 5 | 94.23 | 92.18 | 93.38 |
| 10 | 88.90 | 90.12 | 88.88 |

**TABLE : Accuracy of classifying the validation data using RBF**

**kernel with different and PCA dimensions.(all values in %)**

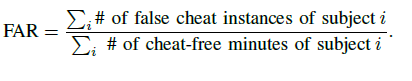
**b) Instance-based metric:** To compute the instance-based metric, we filter the segment-based classification results in the following way. If more than 50%

of the segments, regardless of their relative locations, within a cheating instance are correctly classified as cheating, this is a correctly detected instance. Otherwise, it is a miss detection at the instance level. The TDR in the instance-based metric is defined as:



To evaluate false alarm in the instance-based metric, as long as the number of consecutively detected false cheat segments is over sf , we define this as a falsely detected instance, regardless of its length. Since the instances within the cheat-free portion of the session is not well defined, we compute FAR w.r.t. the

total length (in minutes) of cheat-free videos. Finally, the FAR in the instance-based metric is defined as,



**B. Basic Component Analysis**

We demonstrate the accuracy of the two individual components, text and speech detection, which are the most important ones among all six components. The other components are evaluated along with the entire OEP system in the remaining sections. First of all, we set the parameters used in the six basic components as the following: \_0 = 3, \_v = 0:9, k = 1, d = 0:6 meters, \_g = \_

4 , \_l = 15; 000, \_s = 5; 000 and sm = 50. All experimental results reported

in this section are evaluated with a 5-fold cross-validation on the positive and negative training samples as seen for text, and for speech.

**1) Text detection analysis:** In text detection, the key parameters are the PCA dimensionality and the type of SVM kernel. Different choices of the parameters will affect the text detection performance. Using a two-class SVM [5], Table II

illustrates the detection performance on the validation dataset, with different PCA dimensions and types of SVM kernel reducing the dimensionality does not significantly reduce the detection performance. From this table, we see that the radial basis function (RBF) performs better than other kernels. Since the RBF kernel relies on a good choice of , we tested the detection performance using RBF kernel with different values as seen . It appears that using the SVM with RBF kernel ( = 5) performs best on the validation dataset, where the feature dimension has been reduced to 50. We use these specific parameters in our final OEP system.

**2) Speech detection analysis:** We first analyze the speech detection performance with different acoustic segment lengths Ls. The testing results illustrates that the larger the segment size, the higher accuracy can be achieved. The reason is that the longer audio segment carries more information about speech. However, in a real-world situation the longer the segments are, the more likely the short speech instances will miss detection. To balance between these two cases, we choose the fixed duration Ls as 500ms with a 100ms shift. In order to choose the best kernel, we train the SVM classifiers using different kernels gives the testing accuracy. From this table, we can see that the cubic polynomial function performs best over other kernels. Moreover, we test the performance of SVM using cubic polynomial kernel with different values, and it appears 0:0072 generates the highest accuracy on the testing sound samples.

|  |  |  |  |
| --- | --- | --- | --- |
| Ls | 0.5 | 1 | 2 |
| Accuracy | 95.99 | 98.14 | 99.72 |

**TABLE : Accuracy of classifying audio samples with different**

**Ls lengths.**

|  |  |
| --- | --- |
| **Kernel/Dim** | **Accuracy** |
| Linear | 94.37 |
| Quadratic polynomial | 95.68 |
| Cubic Polynomial | 95.99 |
| Radial basis | 60.65 |
| Sigmoid | 68.17 |

**TABLE : Accuracy of classifying audio samples using SVM with**

**different kernel functions.**

**Chapter 10**

**Plan**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.**  **No.** | **Details of Activity** | **Planned Start Date** | **Planned Complete Date** |
| 1 | **Introduction to Project Topic**: study for selecting  project topic | 06/07/2020 | 07/07/2020 |
| 2 | **Introduction to Project Topic: Discussion** about selected micro-project topic. | 08/07/2020 | 11/08/2020 |
| 3 | **Introduction to Project Topic: Discussion** about selected micro-project topic with concern Mentor. | 13/07/2020 | 16/07/2020 |
| 4 | **Introduction to Project Topic:** Finalize and study for selected topic | 17/07/2020 | 18/07/2020 |
| 5 | **Collect information for studies resource** | 20/07/2020 | 24/07/2020 |
| 6 | **Study for project different resources as literature survey** | 25/07/2020 | 30/07/2020 |
| 7 | **Arrange collected information** | 01/08/2020 | 04/08/2020 |
| 8 | **Making paper for publication** | 05/08/2020 | 08/08/2020 |
| 9 | **Finalizing Paper from concern Mentor** | 10/08/2020 | 13/08/2020 |
| 10 | **Paper Publication in UGL journal** | 14/08/2020 | 14/08/2020 |
| 11 | **Making a presentation related Project Topic** | 17/08/2020 | 17/08/2020 |
| 12 | **Review 1** | 18/08/2020 | 18/08/2020 |
| 13 | **Team Discussion:** Deciding languages, database, techniques, etc | 19/08/2020 | 21/08/2020 |
| 14 | **Presentation**  (for Review 2) | 24/08/2020 | 06/09/2020 |
| 15 | **Code Implementation:**  Completion of Front-end Web System | 24/08/2020 | 02/09/2020 |
| 16 | **Participation in Project competition** | 07/09/2020 | 09/09/2020 |
| 17 | **Project Report** | 10/09/2020 | 15/09/2020 |
| 18 | **Review 2** | 18/09/2020 | 18/09/2020 |

**Chapter 11**

**Conclusions**

Proposed system gives enough idea about design of organizing online test using web-based application and monitor the user throughout the session using different deep learning and computer vision algorithms and libraries. Using Face detection, system can easily check whether user is real or not, also it can check whether one or more people present in the frame. Using eye detector or gaze tracking, we can check that whether the user looking outside the screen or not. Speech to text conversion will check if person is speaking with somebody. Conducting exams online is very important in current time. It enables user to give the test at any time and from anywhere and reduces transportation as well. This will not only save time and cost but it will also create transparency in the examination process.

**Future work**

The future work of this work is to validate the proposed design by implementing the design of continuous user verification in the form of a prototype which will be used to evaluate the method proposed by measuring the verification accuracy and speed. The verification accuracy will be measured by calculating the accuracy rate, FAR and the FRR of the face verification process under various pose and lighting conditions, while the execution time on each verification process will be saved to measure the verification speed.

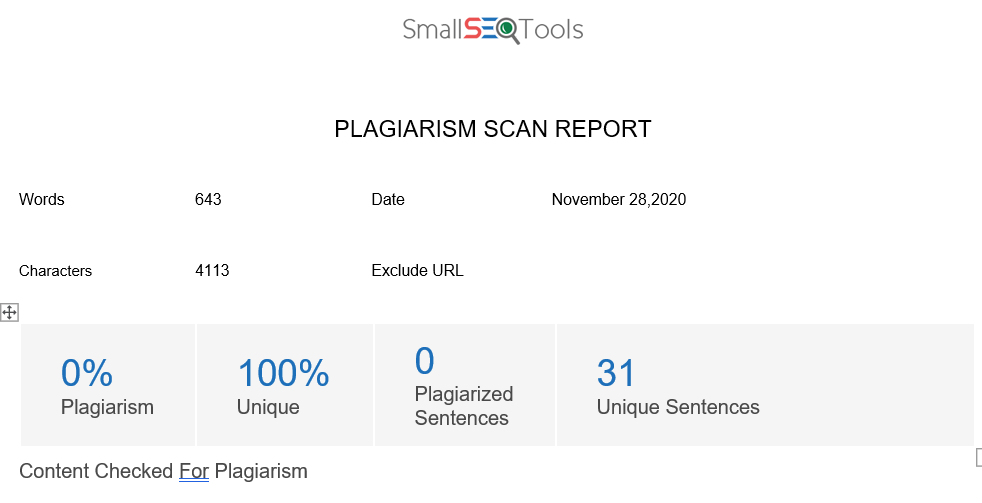
**Chapter 12**

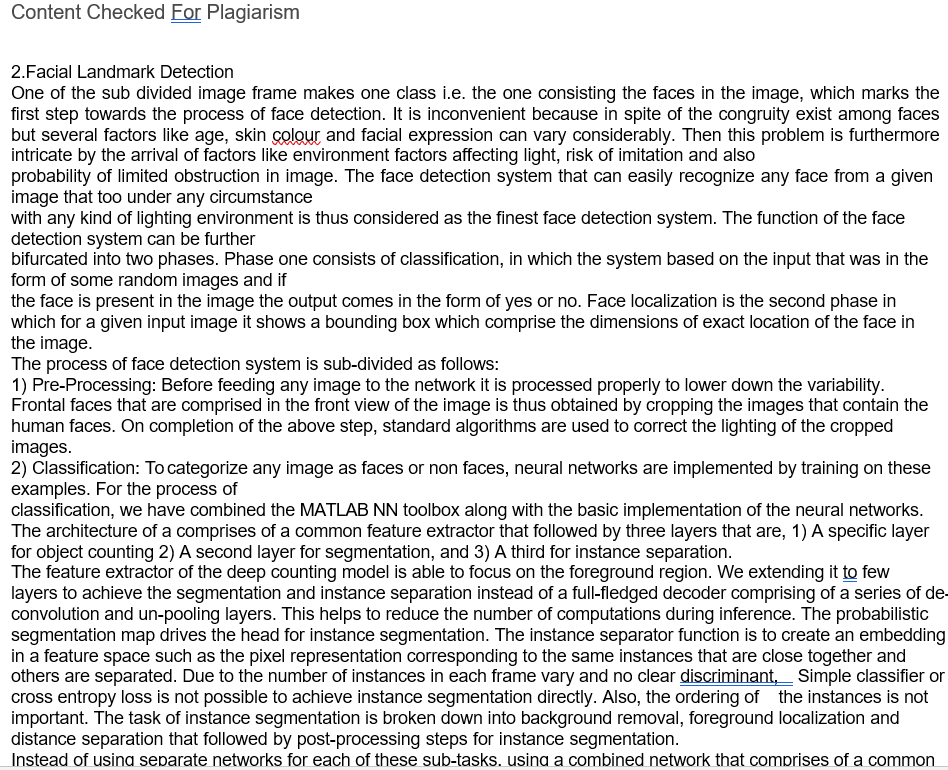
**References**

1. Y. Atoum, L. Chen, Alex X. Liu, Stephen D. H. Hsu, and X.Liu, “Automated Online Proctoring”, IEEE Transaction on Multimedia, 2015.
2. Asep Hadian S. G., Yoanes Bandung, “A Design of Continuous User Verification for Online Exam Proctoring on M-Learning”,2019 International Conference on Electrical Engineering and Informatics (ICEEI)
3. Sanjukta Ghosh, Peter Amon, Andreas Hutter, André Kaup, “DEEP COUNTING MODEL EXTENSIONS WITH SEGMENTATION FOR PERSON DETECTION”, ICASSP 2019
4. Swathi Prathish, Athi Narayanan S, Kamal Bijlani, “An Intelligent System For Online Exam Monitoring”, 2016 IEEE
5. Dweepna Garg. Parth Goel, Sharnil Pandya, Amit Ganatra, Ketan Kotecha, “A Deep Learning Approach for Face Detection using YOLO”, ©20XX IEEE
6. Vishnu R R, Athi Narayanan S, kamalbijlani, “Heuristiv-based Automatic online proctoring system”, 2015 IEEE 15th International Conference on Advanced Learning Technologies
7. Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015.
8. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in Neural Information Processing Systems 25, pp. 1097–1105. 2012.
9. S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, Cambridge, MA, USA, 2015, NIPS’15, pp. 91–99, MIT Press.
10. I. E. Allen and J. Seaman. Grade change: Tracking online education in the united states, 2013. Babson Survey Research Group and Quahog Research Group, LLC. Retrieved on, 3(5), 2014.
11. K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, no. 9, pp. 1904–1916, 2015.
12. J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” CoRR, vol. abs/1506.02640, 2015.
13. V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 12, pp. 2481–2495, 2017.
14. J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3431–3440.
15. Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015.
16. R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” in 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587.
17. C. Szegedy, A. Toshev, and D. Erhan, “Deep neural networks for object detection,” Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2. Curran Associates Inc., pp. 2553–2561, 2013

**Appendix1**

**Plagiarism Report**





**Appendix 2**

**Base papers**

ACCEPTED WITH MINOR REVISION BY IEEE TRANSACTION ON MULTIMEDIA, DEC 30, 2015

# Automated Online Exam Proctoring

Yousef Atoum, Liping Chen, Alex X. Liu, Stephen D. H. Hsu, and Xiaoming Liu

*Abstract*—Massive open online courses (MOOCs) and other forms of remote education continue to increase in popularity and reach. The ability to efficiently proctor remote online examinations is an important limiting factor to the scalability of this next stage in education. Presently, human proctoring is the most common approach of evaluation, by either requiring the test taker to visit an examination center, or by monitoring them visually and acoustically during exams via a webcam. However, such methods are labor-intensive and costly. In this paper, we present a multimedia analytics system that performs automatic online exam proctoring. The system hardware includes one webcam, one wearcam, and a microphone, for the purpose of monitoring the visual and acoustic environment of the testing location. The system includes six basic components that continuously estimate the key behavior cues: user verification, text detection, voice detection, active window detection, gaze estimation and phone detection. By combining the continuous estimation components, and applying a temporal sliding window, we design higherlevel features to classify whether the test taker is cheating at any moment during the exam. To evaluate our proposed system, we collect multimedia (audio and visual) data from 24 subjects performing various types of cheating while taking online exams. Extensive experimental results demonstrate the accuracy, robustness, and efficiency of our online exam proctoring system.

*Index Terms*—Online exam proctoring (OEP), user verification, gaze estimation, phone detection, text detection, speech detection, covariance feature.

## I. INTRODUCTION

M

ASSIVE open online courses (MOOCs) offer the potential to significantly expand the reach of today’s educational institutions, both by providing a wider range of educational resources to enrolled students and by making educational resources available to people who cannot access a campus due to location or schedule constraints. Instead of taking courses in a typical classroom on campus, now students can take courses anywhere in the world using a computer, where educators deliver knowledge via various types of multimedia content. According to a recent survey [1], more than 7.1 million students are taking, at least, one online course in 2013 in America. It also states that 70% of higher education institutions believe that online education is a critical component of their long-term strategy.

Exams are a critical component of any educational program, and online educational programs are no exception. In any exam, there is a possibility of cheating, and therefore, its detection and prevention are important. Educational credentials must reflect actual learning in order to retain their value to society. The authors in [15] state that the percentage of students

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Yousef Atoum is with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI. Liping Chen, Alex X. Liu, and Xiaoming Liu are with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI. Stephen D. H. Hsu is with the Department of Physics and Astronomy, Michigan State University, East Lansing, MI. Corresponding author: Xiaoming Liu, liuxm@cse.msu.edu | | | | |
|  | |  | | --- | | OEP  software | | Cheating  No  Cheating |

Fig. 1: Based on the audio-visual streams captured by a wearcam, a webcam with an integrated microphone, our OEP system *automatically* and *continuously* detects cheat behaviors during online exams.

committing academic cheating activity is on the rise. Nearly 74% of students in 2013 indicated that it would be somewhat easy to cheat in online exams. They also found that in 2013, about 29% of the students admitted to cheating in online exams. When exams are administered in a conventional and proctored classroom environment, the students are monitored by a human proctor throughout the exam. In contrast, there is no convenient way to provide human proctors in online exams. As a consequence, there is no reliable way to ensure against cheating. Without the ability to proctor online exams in a *convenient*, *inexpensive*, and *reliable* manner, it is difficult for MOOC providers to offer reasonable assurance that the student has learned the material, which is one of the key outcomes of any educational program, including online education.

A typical testing procedure for online learners is the following: students come to an on-campus or university-certified testing center and take an exam under human proctoring. New emerging technologies, such as, e.g., Kryterion and ProctorU, allow students to take tests anywhere as long as they have an Internet connection. However, they still rely on a person “watching” the exam-taking. For example, Kryterion employs a human proctor watching a test taker through a webcam from a remote location. The proctors are trained to watch and listen for any unusual behaviors of the test taker, such as unusual eye movements, or removing oneself from the field of view. They can alert the test taker or even stop the test.

In this paper, we introduce a *multimedia analytics* system to perform automatic and continuous online exam proctoring (OEP). The overall goal of this system is to maintain academic integrity of exams, by providing real-time proctoring for detecting the majority of cheating behaviors of the test taker. To achieve such goals, audio-visual observations about the test takers are required to be able to detect any cheat behavior. Many existing multimedia systems [23], [35] have been utilizing features extracted from audio-visual data to study human behavior, which has motivated our technical approach. Our system monitors such cues in the room where the test taker resides, using two cameras and a microphone. As shown in Fig. 1, the first camera is located above or integrated with the monitor facing the test taker. The other camera can be worn or attached to eyeglasses, capturing the field of view of the test taker. In this paper, these two cameras are referred to as the “webcam” and “wearcam” respectively. The webcam also has a built-in microphone to capture any sound in the room. Using such sensors, we propose to detect the following cheat behaviors: (a) cheat from text books/notes/papers, (b) using a phone to call a friend, (c) using the Internet from the computer or smartphone, (d) asking a friend in the test room, and (e) having another person take the exam other than the test taker.

We propose a hybrid two-stage algorithm for our OEP system. The first stage focuses on extracting middle-level features from audio-visual streams that are indicative of cheating. These mainly consist of six basic components: user verification, text detection, speech detection, active window detection, gaze estimation, and phone detection. Each component produces either a binary or probabilistic estimation of observing certain behavior cues. In the second stage, a joint decision across all components is carried out by extracting high-level temporal features from the OEP components at the first stage. These new features are utilized to train and test a classifier to provide real-time continuous detection of cheating behavior. To evaluate the OEP system, we collect multimedia (audio and visual) data from 24 subjects performing various types of cheating while taking a multiple choice and fill in the blank math exam. Extensive experimental results demonstrate the accuracy, robustness, and efficiency of our online exam proctoring system in detecting cheating behavior. This paper makes the following contributions:

* Proposes a fully automated online exam proctoring system with visual and audio sensors for the purpose of maintaining academic integrity.
* Designs a hybrid two-stage multimedia analytics approach where an ensemble of classifiers extracts middlelevel features from the raw data, and transforming them into high-level features leads to the detection of cheating. • Collects a multimedia dataset composed of two videos and one audio for each subject, along with label information of all cheating behaviors. This database is publicly available for future research [[1]](#footnote-1).

## II. RELATED WORK

Over the years, the demand for online learning has increased significantly. Researchers have proposed various methods to proctor online exams in the most efficient and convenient way possible, yet still preserve academic integrity. These methods can be categorized into three categories: (a) no proctoring [7], [34], (b) online human monitoring [8], [13], and (c) semiautomated machine proctoring [17], [24]. No proctoring does not mean that test takers have the freedom of cheating. Instead, cheating is minimized in various ways. In [7], the authors believe they can prompt academic honesty by proposing eight control procedures that enable faculty to increase the difficulty and thus reduce the likelihood of cheating. In [34], the authors offer a secure web-based exam system along with network design which is expected to prevent cheating.

Online human monitoring is one common approach for proctoring online exams. The main downside is that it’s very costly in terms of requiring many employees to monitor the test takers. Researchers have also proposed different strategies in full monitoring, such as in [13], where they use snapshots to reduce the bandwidth cost of transmitting large video files. Authors in [24] attempt to do semi-automated machine proctoring, by building a desktop robot that contains a 360◦ camera and motion sensors. This robot transmits videos to a monitoring center if any suspicious motion or video is captured. The main problem is that a single camera cannot see what the subject sees, and as a result even humans may have a hard time detecting many cheating strategies. For example, a partner who is outside the camera view, but who can see the test questions (e.g., on a second monitor), could supply answers to the test taker using silent signals, or writing on a piece of paper which is visible to the test taker.

Among all prior work, the most relevant work to ours is the Massive Open Online Proctoring framework [17], which combines both automatic and collaborative approaches to detect cheating behaviors in online exams. Their hardware includes four components: two webcams, a gaze tracker, and an EEG sensor. One camera is mounted above the monitor capturing the face, and the other is placed on the righthand side of the subject capturing the profile of the subject. Motion is used for classification by extracting dense trajectory features. However, this work is limited to only one type of cheating (i.e., reading answers from a paper), with evaluation on a small set of 9 subjects with 84 cheat instances. Since many types of cheating do not contain high-level motion, it is not clear how this method can be extended to handle them. To the best of our knowledge, there is no prior work on a fully automated online proctoring system that detects a wide variety of cheating behaviors.

Beyond educational applications, in the multimedia community, there is prior work on audio-visual-based behavior recognition. Authors in [35] study audio-visual recordings of head motion in human interaction, to analyze socio-communicative and affective behavioral characteristics of interacting partners. [21] automatically predicts the hireability in real job interviews, using applicant and interviewer nonverbal cues extracted from the audio-visual data. In [10], they automatically estimate high and low levels of group cohesion using audiovideo cues. In [16], the authors use audio-visual data to detect a wide variety of threats and aggression, such as unwanted behaviors in public areas. Their two-stage methodology decomposes low-level sensor features into high-level concepts to produce threat and aggression detection. While there is similarity between their methodology and ours, our unique two-camera imaging allows us to leverage the correlation between the two distinct visual signals. The addition of audio to video was also proven to complement many visual analysis problems, such as object tracking [14], event detection retrieval in field sports [27], and vision-based HCI system [23].

|  |
| --- |
| Fig. 2: The architecture of the Online Exam Proctoring (OEP) system. |

One of our novel ideas is to use a second wearcam for capturing the full field of the view of the subject. This is similar to the research in first person vision where visual analysis is performed on the wearcam. For example, [30] temporally segments human motion into actions and performs activity classification in the context of cooking. [31] uses a wearcam to detect the iris and estimate the visual field in front of the subject, which helps to identify where exactly the subject is looking. In contrast to the single wearcam in the first person vision, our OEP system utilizes two cameras to capture both what the subject sees and his/her own behavior, which enables comprehensive behavior profiling.

## III. PROPOSED METHOD

In this work, we aim to develop a multimedia analysis system to detect a wide variety of cheating behaviors during an online exam session. Our proposed online exam process includes two phases, the *preparation phase* and *exam phase*. In the preparation phase, the test taker has to authenticate himself before beginning the exam, by using a password and face authentication. This phase also includes calibration steps to ensure that all sensors are connected and functioning properly. Further, the test taker learns and verbally acknowledges the rules of using the OEP system, such as, no second person is allowed in the same room, the test taker should not leave the room during the exam phase, etc.

In the exam phase, the test taker takes the exam, under the continuous “monitoring” of our OEP system for real-time cheating behavior detection. As shown in Fig. 1, we use three sensors (i.e., webcam, wearcam and microphone) to capture audio-visual cues of the exam environment and the test taker. The sensed data is first processed using six components to extract middle-level features as seen in Fig. 2. These components are: user verification, text detection, speech detection, active window detection, gaze estimation, and phone detection. After that, the middle-level features within a temporal window are fused to generate high-level features, which are then used for training and testing a cheat classifier. The high-level features include the component-dependent features, such as the mean and standard deviation within a window, and features based on the correlation among the components, such as the covariance features [32]. It is crucial to use a diverse and rich set of features to improve the overall detection performance of the OEP system, since the detection of some cheating behaviors relies on the ignition of multiple behavior cues.

The remainder of this section describes the following topics:

(A) the hardware components of the OEP system, (B) through (G) the six basic components of the system, and (H) the highlevel features and classification of the cheating behavior.

### A. Hardware Components

During an exam, the test taker may cheat by *hearing* or *viewing* forbidden information. Therefore, the OEP system hardware should be designed in a way to hear what the test taker hears and see what the test taker sees. This leads to our design of three hardware components: a webcam, a wearcam, and a microphone. The webcam is mounted on top of the monitor facing the test taker and serves multiple purposes, e.g., knowing who is the test taker, what is he doing, and where is he looking. The wearcam is a wearable camera intending to be attached to the test taker’s head, such that the camera is pointing to the same pose direction as the face. Since the wearcam essentially captures the field of view of the test taker, analyzing its video content enable us to detect the “viewing-based” cheating behaviors, such as reading from books, notes, papers, and smartphones. The wearcam contributes significantly in estimating the head gaze, which is an important behavior cue. Note that employing the wearcam is a distinct novelty of our system design, as well as an advantage over prior exam proctoring systems. This design is not only motivated by the need to see what the test taker sees, but also the growing popularity and decreasing cost

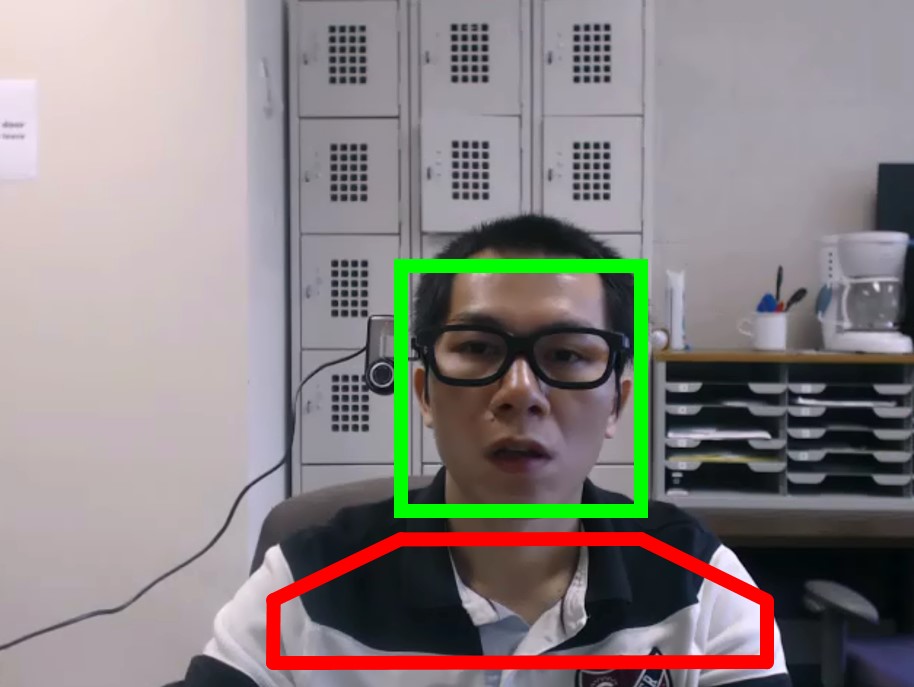


Fig. 3: The extracted region of the face (green) and the body (red).

of wearable cameras. Finally, as an integrated device of the webcam, the microphone captures what the test taker hears based on our rules, any detected human voice is considered as potential cheating.

During the system design, we experimented to find a suitable prototype for the wearcam. We initially tested the system with a Sony action cam by utilizing a headband. However, the relatively heavy weight and the need to synchronize the webcam and wearcam made this option undesirable. We finally decided to attach a regular wired webcam to a pair of eyeglasses, considering the fact that webcams are becoming smaller in size, lighter in weight, cheaper over the years, and have real-time wireless capabilities. Similar ideas have also been adopted in the research community to understand human behavior [30], [31]. Note that our OEP system does not depend on a specific choice of cameras, if a more suitable wearcam is available in the future, we can easily adopt it in our system.

Both cameras capture video at a resolution of 640 × 480 and a frame rate of *fs* = 25 fps. Since the OEP system starts to grab video streams from two cameras at the same time, the two video and audio streams are automatically synchronized during the test session.

### B. User Verification

One of the major concerns in online exams is that the test taker solicits assistance from another person on all or part of an exam. An OEP system should be able to continuously verify whether the test taker is who he claims to be throughout the entire exam session. The test taker is also expected to take the exam alone without the aid of another person in the room. While there are various options for continuous user authentication, such as keystroke dynamics, we decide to use face verification due to its robustness.

There are a number of challenges for user verification in OEP. First, face detection under various lighting and poses is difficult. Second, due to the partial occlusion caused by the eyeglasses with the attached wearcam (Fig. 2), the performance of face detection and verification can be more fragile. Finally, although face detection has improved substantially over the years, occasional miss detections and false alarms are inevitable, and how to handle this is another challenge.

We propose to overcome these challenges by using an approach integrating both face and body cues. We use the Minimum Average Correlation Energy (MACE) filter to perform face verification [28]. During the preparation phase, initial face authentication is conducted by matching the webcam-captured

|  |
| --- |
| Algorithm 1: User verification algorithm. |
| Data: A new frame I*t*, h*f*  Result: v*p*, v*n*  Initialization: *v* = 0, *c*0 = *c*1 = 0 ; Viola-Jones face detector → v*n*(*t*) ; switch *vn*(*t*) do  if *c*0 *> τ*0 then *pt* = v*p*(*t*) = *c*0 = 0; % warning is sent  case  0  else  case  1  case  *>*  1  else  *c*0 ++; v*p*(*t*) = v*p*(*t* −1);  *pt* = *F*(v*p*(*t*)*,v,t,*¯ *p*¯);  *c*0 = *c*1 = 0; if *v* = 1 then  Compute h*t*, *pb* = h*Tt* h*b*;  if *pb > τv & pt*−1 *> τv* then *pt* = *F*(v*p*(*t*)*,v,t,*¯ *p*¯);  else  *v* = 0;  if *v* = 0 then c*t* = x*t* Nh*f*, v*p*(*t*) = PSR(c*t*);  *pt* = *F*(v*p*(*t*)*,v,t,*¯ *p*¯); if *pt > τv* then *v* = 1, *t*¯= *t*, *p*¯= *pt*;  if *c*1 *> τ*0 then *pt* = v*p*(*t*) = *c*1 = 0; % warning is sent  *c*1 ++; v*p*(*t*) = v*p*(*t* −1);  *pt* = *F*(v*p*(*t*)*,v,t,*¯ *p*¯); |

faces with a mugshot of the test taker. In the meantime, a set of frontal-view images of the test taker is captured, where we detect the faces via the Viola-Jones face detector [33], and train a MACE Filter h*f*. As shown in Fig. 3, from the body region of the images, we extract a 160-dim HSV color histogram of the clothing h*b*. The body region has a width equal to twice the width of the detected face, and a height equal to half the height of the face. During the exam phase, when a new frame **I***t* is captured by the webcam, we first perform face detection. Depending on the number of detected faces v*n*(*t*), we handle it correspondingly, as described in Algorithm 1.

If only one face is detected in the new frame (v*n*(*t*) = 1); this is the most likely case since the test taker is required to take the exam alone. Let x*t* be the appearance feature of the detected face, *pt* be the probability of user authenticity, and *v* be an indicator flag on whether the test taker is verified. If the user is not verified (i.e., *v* = 0) in the previous frame, we verify the user by performing cross-correlation c*t* = x*t* Nh*f*, where c*t* is the correlation output at time *t*. For computational efficiency, correlation is computed in the Fourier domain using Fast Fourier Transform (FFT), and then transformed back to the spatial domain via inverse FFT. Savvides et al. showed that c is sharply peaked for authentic subjects, and does not exhibit a strong peak for impostors [28]. The Peak-to-Sidelobe Ratio (PSR) is defined to measure the strength of the correlation peak, where a PSR value greater than five is considered as an authenticated user. We denote the PSR value computed at time *t* as v*p*(*t*), which is further converted into the probability

|  |
| --- |
| Algorithm 2: Face verification probability *F*. |
| Data: v*p*(*t*)*,v,t,*¯ *p*¯ Result: *pt*  if *v* = 0 then if *vp*(*t*) *<* 5 then *pt* = 0;  else  else if *vp*(*t*) > 10 then *pt* = 1;  else  ;  *pt* = ¯*pe*−*k*(*t*−*t*¯); |

measure of user authenticity *pt*, by using the function *F*,

*pt* = *F*(v*p*(*t*)*,v,t,*¯ *p*¯)*,* (1)

as explained in Algorithm 2. If *pt* is larger than a predefined threshold *τv*, the face is verified, and we denote *t*¯ the last verified time and *p*¯ the last verified probability. Otherwise, the face continues to be verified in the next frame.

When the next frame arrives and only one face is detected, if the user is verified before (*v* = 1), we rely on body tracking due to its robustness to head poses, instead of face verification. Specifically, we compute the histogram of the clothing h*t*, and compare it to h*b*. If their similarity *pb* is larger than a threshold *τv*, *pt* is calculated as *pt* = ¯*pe*−*k*(*t*−*t*¯), where *k* is the decay speed of the exponential function. After ∆*t* = *t* − *t*¯ seconds from the last verification time, the face needs to be verified again even if *pb > τv* all the time. This is reasonable because an impostor could wear the same clothes as the test taker.

There are cases where no face is detected in the current frame (v*n*(*t*) = 0). When the number of consecutive frames without detected faces, *c*0, is bigger than a threshold *τ*0, the system determines that the user has left the exam and a warning is sent with an assigned high probability of cheating. Face verification is required to continue the exam when the user appears again. If *c*0 ≤ *τ*0, we do not make any decision and wait for the next frame. This tolerance is necessary because the face might not be detected in certain scenarios, e.g., the large pose, illumination changes or occlusion.

If more than one face is detected (v*n*(*t*) *>* 1), we also consider some tolerance, similar to the case of v*n*(*t*) = 0. When the number of consecutive frames with multiple detected faces, *c*1, is less than *τ*0, we do not make any decision and wait for the next frame. Otherwise, there is indeed more than one person in front of the computer. A warning is sent, with a high probability of cheating.

The user verification component provides continuous estimation per frame regarding the number of faces and PSR values, which are stored in two vectors, “numFaces” and “facePSR”, respectively. The numFaces vector, v*n*, is a direct indication of cheating when v*n*(*t*) 6= 1. However, the facePSR alone, v*p*, may only implicitly represent cheating. This output will be converted to high-level features to serve in the process of detecting other cheat behaviors as seen in Fig. 2.

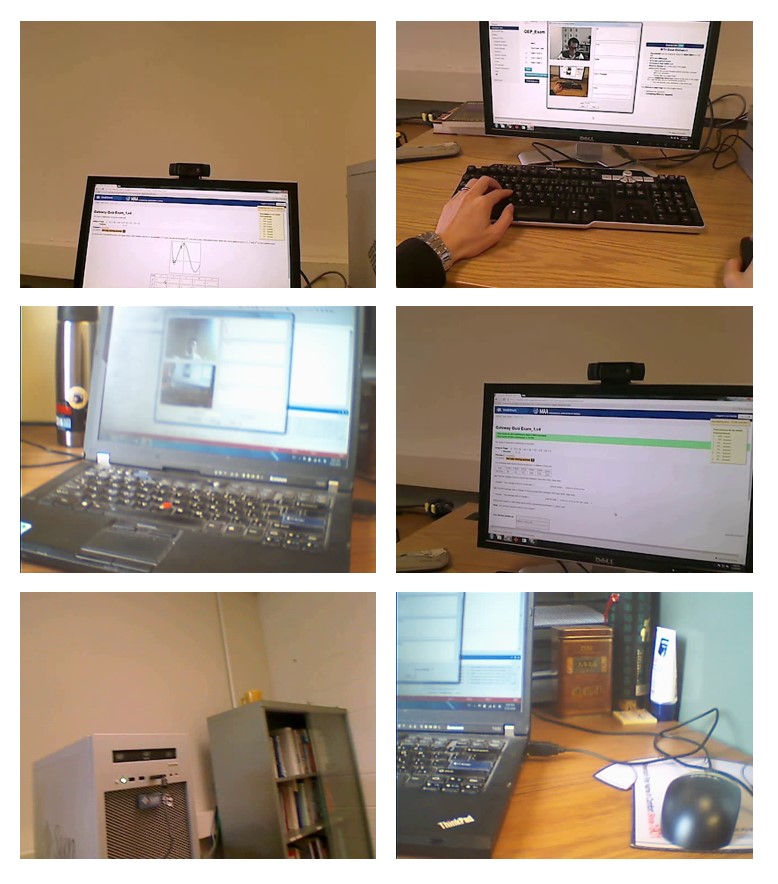
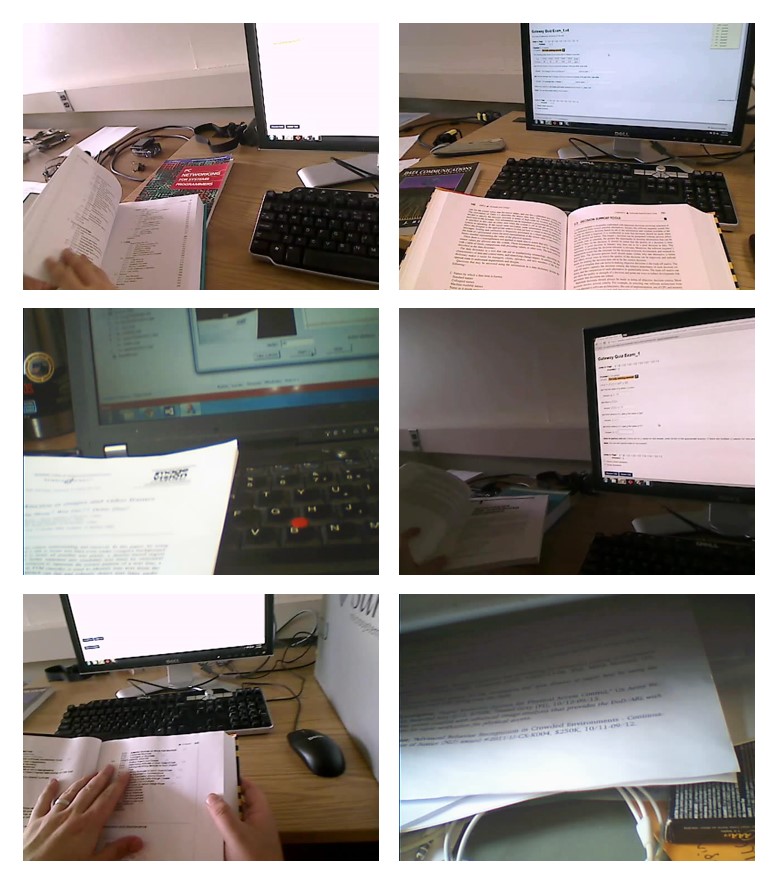


Fig. 4: Positive (left) and negative (right) samples for text detection.

### C. Text Detection

In a closed-book exam, reading from text is a major form of cheating, where the text can be from a book, printout, notes, etc. It is obvious that the webcam alone cannot effectively detect this cheat type since the webcam might not “see” the book or printout. On the other hand, the wearcam captures everything in the field of view of the test taker. Hence, any text seen by the test taker can very likely be seen, and detected, through the wearcam.

While text detection is a well-studied topic, detecting text in online exams could be challenging, since the test taker may attempt to cheat from text with small font, or place the text far away from the camera. Further, we need to differentiate text on printed papers vs. the text on the computer screen or the keyboard, since detection of the latter is not considered as cheating, as shown in Fig. 4. Note that for this work, we focus on printed text only, rather than handwriting. In the case of handwriting, the aid of other capabilities might be needed, such as estimating the eye gaze of the user, since cheating from text requires the test taker to look at it for some time. Moreover, motion blur could also be introduced due to fast head movements. In such cases, a motion blur detector would be employed and then we can skip text detection on these frames with blurred motion.

We develop a learning-based approach for text detection. First, we collect a set of 186 positive training images that contain text in a typical office environment, and 193 negative training images (Fig. 4). Then a learning algorithm based on the GIST features [22] is applied to the training images. We perform cross-validation to estimate the algorithm’s parameters, and finally, the algorithm can predict the probability of text in a testing video frame.

The GIST feature is well known in the vision community. For example, [22] introduces how to compute GIST features, based on a low dimensional representation of a given image, termed “Spatial Envelope”. A set of perceptual dimensions (naturalness, openness, roughness, expansion, ruggedness) that represent the dominant spatial structure of an image is used. Since the GIST features of images are 512-dimensional vectors, we apply PCA to reduce them to a lower dimension, which are then used for training a binary SVM classifier. Given a testing video frame, the output of the SVM classifier is stored as one element of the “textProb” vector, denoted by

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sound type | # files | Length | Sound type | # files | Length |
| speech | 4 | 154 | keyboard typing | 7 | 18 |
| burp | 3 | 2 | key jingle | 5 | 21 |
| chair moving | 1 | 8 | paper moving | 5 | 14 |
| cough | 5 | 7 | phone ring | 7 | 16 |
| door knocking | 6 | 8 | runny nose | 3 | 5 |
| open/close door | 4 | 4 | sigh | 8 | 10 |
| drink | 7 | 15 | silence | 2 | 9 |
| fart | 4 | 6 | breath | 5 | 33 |
| gasp | 4 | 4 | spit | 3 | 4 |
| hiccup | 5 | 2 | steps | 6 | 25 |

TABLE I: Collected sound samples, with the total number and duration length (in seconds) of sound files.

v*t*, representing the probability of detecting text in a frame.

### D. Speech Dectection

One of the most likely cheating behaviors in online exams is to seek verbal assistance from another person in the same room, or remotely via a phone call. In fact, from the audiovisual dataset collected in our work, this is the most frequent cheating behavior. By requiring the test taker to take the exam in a quiet room with no one around, any human speech being detected could be considered a potential cheating instance. Therefore, we design algorithms in this component to detect speech from acoustic signals.

There are unique challenges for speech detection in OEP. Test takers who attempt cheating tend to use a low voice while speaking to others. Therefore, one challenge is to be able to detect speech at any level of amplitude. Second, speech can be confused with many environmental sounds in the test room, such as noises generated from moving objects (e.g., chair, door, or keyboard), while others might be caused by the test taker, e.g., coughs or breathing. This can be especially challenging when speech is overlaid with other sounds.

Following a learning-based speech detection scheme, we first collect a wide variety of typical sounds in an office environment, such as breath, burp, chair moving, cough, steps, etc. Table I shows the number of files and the length of the audios for each sound category. These sounds are either found online [4] or recorded by ourselves. We only consider speech as the positive samples, while the remaining categories of sound are negative. In total, the lengths of positive and negative samples are 154 and 211 seconds, respectively.

Unlike text detection where the unit of classification is an image, the unit of speech detection is an acoustic segment. A segment is defined either when the amplitudes of all its samples are larger than a threshold, or with a fixed duration *Ls*. Due to its simplicity and robustness, we decide to adopt the latter approach. It is a trade-off to determine the length of *Ls*, as the longer duration leads to a higher detection rate for detecting long speech, but a lower rate for shorter speech.

The acoustic segment is represented by the short-time Fourier transform (STFT) using Hamming windows [9]. We divide the frequencies from 200 Hz to 4 KHz into 16 different channels. Then we extract a 138-dimensional feature, which encodes the mean and standard deviation of the power percentile in each frequency channel and of the total power, bandwidth, the most powerful frequency channel, the number of peaks in power over time, the regularity of power peaks, the range of the total power over time, and time-localized frequency percentiles over various frequency ranges.

With the collection of features from training samples in Table I, we use a binary SVM classifier for speech detection. During testing, the output of the SVM classifier is stored as one element of the “voiceProb”, denoted by v*v*, representing the probability of detecting speech within a sound segment.

### E. Active Window Detection

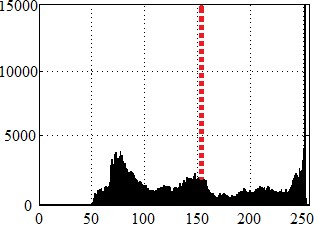
The Internet and computers are an open gateway to valuable information for answering exam questions. The authors in [15] indicate that cheating from the Internet is the most frequent among e-learners. In [7], they use Blackboards Respondus Lockdown Browser (RLB) to access the online exam. RLB is a special browser where the test taker is locked into the exam and has no way to exit/return, cut/paste, or electronically manipulate the system. However, some exams might require Internet access to some specific websites, or perhaps the use of e-mail or chat functions. Moreover, some test takers might have saved files and documents on the computer containing answers to the exam. Therefore, it is critical to keep track of how many windows the test taker is opening.

In our OEP system, we give the user full Internet and computer access during the exam. We periodically estimate the number of active windows running in the system, denoted by v*w*, obtained from the operational system API. Most of the time, there should be only one active window, which is the online exam itself. If v*w*(*t*) *>* 1 at a specific time *t* during the exam, we assume the test taker is cheating, and a warning will be displayed on the monitor requesting an immediate shutdown of the opened window. The probability of cheating increases as the test taker keeps the unexpected window opened longer. Since this component relies on the operational system API, the accuracy of active window detection is 100%.

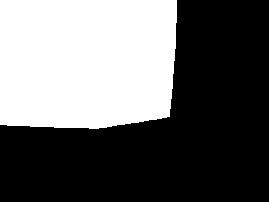
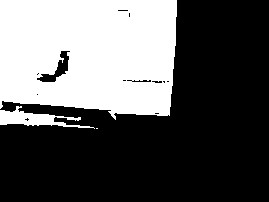
### F. Gaze Estimation

In traditional classroom-based proctoring, the abnormal head gaze direction and its dynamics over time can be a strong indicator of potential cheating. For example, an abnormal gaze is when the test taker’s eyes are off the screen for an extended period of time, or if the head quickly gazes around a few times. Although abnormal gaze does not directly constitute a cheating behavior, it is an important cue to suggest the potential subsequent cheating actions.

As a classic computer vision problem [19], head gaze estimation is a particularly challenging problem in our application due to the spontaneous head motion of the test taker as well as the partial occlusion by the eyeglasses and wearcam. To address this issue, we take advantage of both visual sensors to enhance head gaze estimation. From the wearcam, gaze can be inferred based on the relative 2D location of the monitor screen. From the webcam, we may estimate the gaze from the face in the video frame. By combining the information from both cameras, we accurately estimate the head gaze of the test taker in a wide range of yaw and pitch angles.



(a) (b) (c)



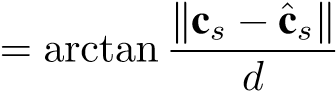
(d) (e) (f)

Fig. 5: Screen detection process: (a) input frame I*t*, (b) grayscale image, (c) histogram of (b), (d) converted binary image based on the threshold, (e) the largest region after connected component analysis, and (f) estimated screen using the convex hull of the largest region.

We now describe the gaze estimation from the wearcam, where the core routine is to extract the position of the screen automatically. We achieve this based on a simple observation that the pixels of the screen are brighter than other pixels. Specifically, as seen from Fig. 5(a-d), we first convert the image to grayscale, and then to binary by using a proper threshold, which is set to the mean intensity of the grayscale image. Using connected component analysis and only keeping the largest region, we obtain a candidate region of the screen. Finally, the screen is extracted by computing the convex hull of the large region.

In the preparation phase, the user is required to be in frontal view of the webcam, while performing initial authentication. As a result, it is reasonable to assume that the screen is near the center of the video frame from the wearcam. We indeed verify this before completing the preparation phase. In order to use the screen position to estimate the head gaze in the exam phase, we calibrate the screen position during the preparation phase. That is, we estimate the screen position, and denote its center as c*s*, width as *ws*, and height as *hs*. Note that calibrating the screen is also very important for other components, such as the text and phone detection. We also learn an HSV model of the screen consisting of two thresholds, an upper and lower bound of possible screen intensity across the color channels. The bounds are defined by the mean and standard deviation of each channel in the preparation phase. Using this model, in the exam phase, an HSV pixel is converted to foreground (i.e., 1 in the binary image), if and only if all the H, S and V intensities fall within the learned bound.

During the exam phase, given a new frame, we use the HSV model to convert the frame to a binary image and then estimate the screen position ˆc*s*. We assume the distance between the test taker and the screen is set to a fixed distance of *d*. Knowing *d*, c*s* and ˆc*s*, the head pose is calculated by

v*g* *.* (2)

It is obvious that we may only estimate v*g* using the screen region when the screen is visible in the wearcam video. That is, when the head gaze is larger than *θg*, the screen is out-ofview from the wearcam video frame. In this case, we use the second approach of head gaze estimation via the face image captured by the webcam.

The basic idea of this second approach is similar to the approach in [3]. At the initial step, we detect a set of strong corner points on the face [29], and then convert them to 3D model points by using a sinusoidal model. This model attempts to map the 2D corner points on a 3D sinusoidal surface, which is an approximation of the true 3D face surface. Secondly, we track these points by using the Lucas-Kanade method, and estimate a rotation matrix based on the changes of the tracking points. We observe that at small gaze angles, the screen-based approach is superior to the face-based approach. Therefore, the face-based approach is only utilized when v*g > θg*.

For each frame, we store the results of the gaze estimation into elements of two vectors, “gazeLR” and “gazeUD”, which are denoted as v*g*1 and v*g*2, respectively. The first represents the yaw estimation, and the second is the pitch estimation.

Since the estimated gaze is an angular value in the range of

, we normalize them such that v*g*1 ∈ [−1*,*1], where −1 means the user is looking far left at an angle of  and 1 is towards the far right at. The same applies to v*g*2.



### G. Phone Detection

Our online exam rule prohibits the use of any type of mobile phones. Therefore, the presence of a mobile phone in the testing room can be an indication of potential cheating. With advancements in mobile phone technology, there are many ways to cheat from them, such as reading saved notes, text messaging friends, browsing the Internet, and taking a snapshot of the exam to share with other test takers.

Phone detection is challenging due to the various sizes, models and shapes of phones (a tablet could also be considered a type of phone). Some test takers might have large touch screens while others might use a button-based flip phones. Moreover, cheating from a phone is usually accompanied with various occlusions, such as holding the phone under the desk, or covering part of the phone with their hand.

To enable this capability, we utilize the video captured from the wearcam, since it sees what the test taker is seeing. We perform phone detection based on a similar approach for screen-based gaze estimation, i.e., searching for pixels that are brighter than the background pixels. The motivation of using the screen’s brightness over detecting the phone object, is that we don’t want to claim there is a phone-based cheating behavior unless the phone is switched on. By using additional constraints on the area of potential local regions to exclude large (i.e., the monitor) and small (i.e., random noise) objects, whose thresholds are denoted as *τl* and *τs* respectively, we can estimate a candidate local region for the phones screen. We chose to represent the estimated phone screen by using the area of the local region.

Given a video frame from the wearcam, the output of the phone detection model is stored as one element of the “phoneProb” vector, denoted by v*ph*. Since the phone detection module detects phone with an area in the range of [*τs,τl*], we normalize them such that v*ph* ∈ [0*,*1], representing the probability of detecting a phone in the frame. Since the vector

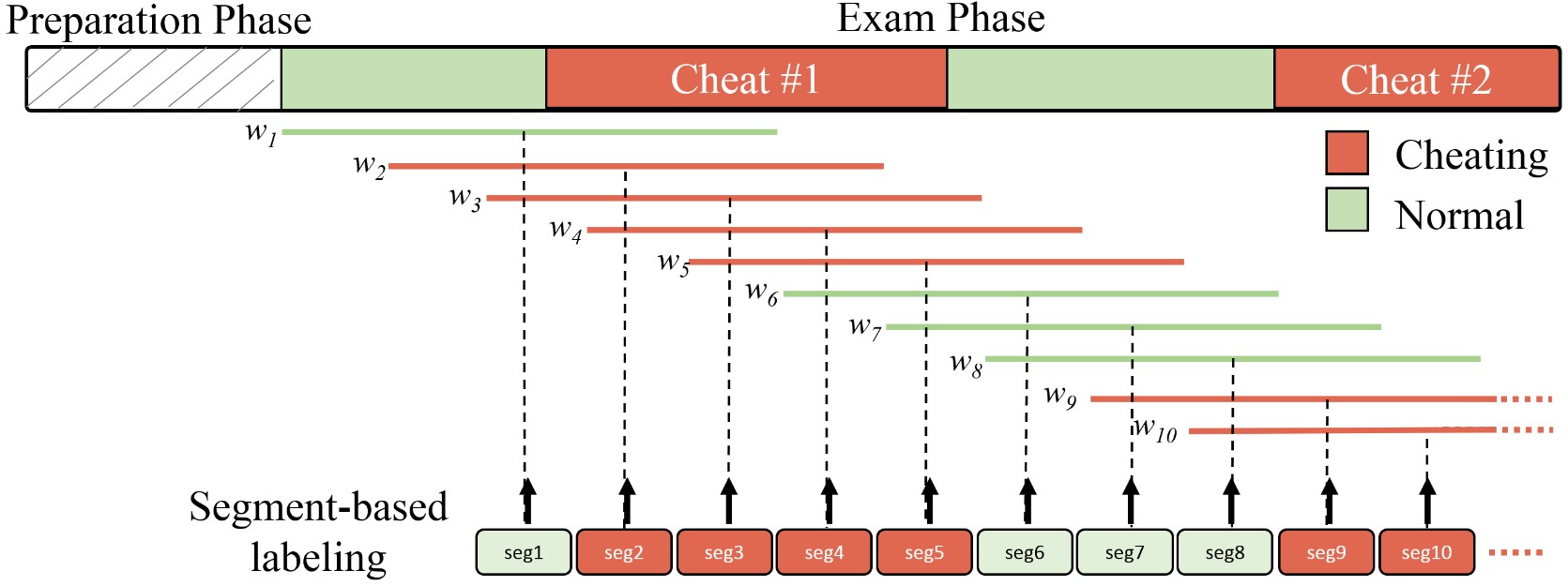


Fig. 6: Segment-based labeling process for a subject. In this example, the test taker cheats two times during the exam. Note how the window *w* shifts at exact increments with an 80% overlap. At each shift, a segment is formed and assigned a label based on the majority vote of ground truth labels that falls within *w*.

v*ph* could be noisy, we apply a median filter of a fixed size *sm*, to eliminate the random noise.

### H. Cheating Behavior Detection

At this stage, we have the continuous output of the OEP basic components (i.e., v*p*,v*t*,v*v*,v*g*1,v*g*2,v*ph*,v*w*,v*n*), where all vectors have the same sampling rate, i.e., one element per frame. We now present how to further analyze these vectors to detect cheat behaviors. Note that, as seen by the blue dashed arrows in Fig 2, the latter two vectors v*w* and v*n* (i.e., number of active windows and faces), are used directly to provide a cheat decision. On the other hand, the remaining six vectors will be utilized for extracting high-level features, which will then be used for learning a SVM classifier to make continuous decisions on cheat behaviors.

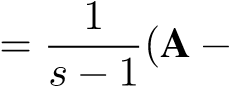
In our algorithm design, we highlight the correlation among the multiple components, which is extremely valuable in detecting many cheat behaviors. For instance, it is shown that when test takers cheat by talking to a person in the test room, there is a high correlation between the gaze and speech estimation, which means that the test taker tends to look at the person during this process. Another example is between the gaze and text detection, where the subjects tend to turn left or right to search for a book or some notes. We now explain how we design these high-level features, and the cheat classifier used in the OEP system, in the following two subsections.

1. *Feature extraction:* Since cheating behaviors occur over a time duration, features need to be defined based on the temporal window, which is commonly adopted in other behavior recognition work [2]. We define a temporal window *w* with a fixed length of *s* seconds for the purpose of feature extraction. By shifting the window throughout the middlelevel feature vectors with a fixed overlap of *l*, we generate multiple segments, which are the units for both training and testing. Given that we manually label the ground truth (cheat vs. non-cheat) for all collected videos at each second, we can convert this labeling to the ground truth label of each segment. That is, the binary ground truth label of a segment is determined by the mass majority of per-second ground truth labels within a segment. The window length *s* is preferred to be exact integer seconds, as well as an odd number of seconds to remove potential equality. The temporal segmentation and labeling process are illustrated in Fig. 6.

At time *t*, the high-level features are extracted from all six vectors within the temporal window *wt*, and used to represent the segment. The high-level features of each segment are composed of the mean *µ*, standard deviation *σ* of each component vector, and the covariance features C.

The covariance feature is an effective visual feature used in many vision systems, including pedestrian detection [32]. Let v*i* be the *ith* component vector obtained from one segment. We compute a *sfs* × 3 matrix A*i* = [v*i* |v0*i*| |v00*i* |], where

|v0*i*|*,*|v00*i* | are the absolute values of the first and second order derivatives, respectively. Due to the sparsity of v*ph* (i.e., most elements of the vector are zeros as seen in Fig. 2), we exclude it from extracting covariance features. Therefore, combining A*i* of all the remaining five vectors yields a *sfs* × 15 matrix, A = [A1A2*...*A5]. To compute the covariance feature, we apply the following equation:

C  mean(A))*T*(A − mean(A))*,* (3)

where mean() computes the mean across all rows. Since C is a 15×15 symmetry matrix, by keeping the upper triangular, the covariance feature of a segment is a 120-dimensional vector. Finally, each extracted segment has a 132-dimensional feature, including the *µ*, *σ* (6 dimension each obtained from the 6 basic components), and the covariance feature (obtained from 5 basic components excluding the phone detection), to be used for cheat classification.

1. *SVM cheat classifier:* As with the OEP components, we use SVM for classifier learning [5]. For all training videos in the OEP dataset, the segments with no cheating are considered as samples of the negative class, and the rest segments are of the positive class. We divide the positive cheating samples into three main categories. (a) Any text related cheating from books, papers and notes is assigned to class 1. (b) Any cheating involving speech such as asking a person in the room, calling a friend on a phone, or any other speech detected in the room, is assigned to class 2. (c) Cheating from a phone or laptop device is assigned to class 3. Class 0 is reserved for the no cheating segments (the negative class). It is observed that a multi-class SVM, consisting of a set of three pair-wise binary classifiers (class 0 vs. 1, 0 vs. 2, etc.), performs better than the binary classifier (class 0 vs. class 1*,*2*,*3) During the testing, we feed the feature of each segment to three classifiers, and use the average of the three classification scores as the final measure of the cheating likelihood.

## IV. OEP DATABASE COLLECTION

Since there is no publicly available database for online exams, we carefully designed a protocol for data collection and labeling. The data collection took place in a room with regular office furniture. We prepared a mathematics online exam consisting of several multiple choices and fill in the blank questions as shown in Fig. 7. During the preparation phase of the exam, we inform the test taker of a set of rules they need to obey: (a) No books, notes or any sort of text are allowed in the room. (b) Phones and laptops are prohibited. (c) The student has to solve the problems without the help from any other person. (d) Using the Internet is prohibited.



|  |
| --- |
| Fig. 8: OEP dataset examples illustrating various cheat types. The examples are grouped in pairs showing both webcam and wearcam at a specific time of the exam. The subjects are cheating from books, notes, papers, smartphones, the Internet, or asking someone in the room. |

Fig. 7: Two questions of the mathematics exam that are given to test takers during data collection.

A total of 24 subjects, all of whom are students at Michigan State University, participated in the data collection. The first 15 subjects were actors that pretended to be taking the exam. They were asked to perform cheating behaviors during the session, without any instructions on what cheating behavior to perform or how to perform them. One issue with these subjects is that potentially artificial behaviors are observed during the acting. Therefore, to capture real-world exam scenarios, we asked nine students to take the real exam, where their scores were recorded. Knowing that they are not likely to cheat in the data capturing room, the proctor invokes the cheating behaviors by talking, walking up to the student, or handing them a book, etc. The combination of these two types of subjects enriches the database with various cheat techniques, as well as the sense of engagement in real exams.

For each of 24 sessions, we collect the audio and two videos from both cameras as seen in Fig. 2. Each session varied in length with an average time of 17 minutes. Human annotation and labeling are performed offline after collecting the data by viewing the two videos and audio simultaneously. The labeling of one cheat instance consists of three pieces of information: the start time, end time and type of cheating. We label five different types of cheating behaviors: (1) cheating from a book, notes or any text found on papers. (2) talking to a person in the room. (3) using the Internet. (4) asking a friend a question over the phone. (5) using a phone. The labeling process for every session is done carefully and required nearly 30∼35 minutes per session. Fig. 8 illustrates examples of different types of cheating from various subjects.

Nearly 20% of the total video length has various cheating activities while the remaining 80% contains normal exam tak-

Cheat type frequency Total cheat duration per type

4%

34

%

50

%

10

%

3

%

%

16

50

%

%

21

%

1

%

12

Read text

Ask friend

Search Internet

Call friend

Search Phone

Fig. 9: Statistics of cheating behavior in the OEP dataset. Cheat types: (1) cheat from book/note/paper, (2) talk in room with a person, (3) use the Internet, (4) ask a friend over the phone, and (5) use a phone or other devices.

ing behaviors with no cheating. Even though these percentages may not depict real life exam scenarios (e.g., 1% cheat vs. 99% normal), it is necessary for the OEP system to include as many cheating instances as possible to learn and evaluate a cheat classifier. Fig. 9 shows a full description of the cheat behaviors in our OEP dataset. The total duration of all types of cheating is reported to be 7*,*235 seconds. The most frequent cheat behavior is type 2 then type 1, summing up to a total of 84% of all cheat activities. The total number of cheat behaviors performed by all subjects is equal to 569 instances, varying in the type and duration of cheating.

The five cheat types defined in our system cover all kinds of cheating behaviors we could manually identify in the collected OEP dataset. It is reasonable to assume that they are also the most common cheating techniques in the real world. Note that the techniques used within a specific type can vary from one subject to another, increasing the level of difficulty in detecting some of the instances. For example, some students may open a book in front of them to cheat from, while others hide the book behind the computer screen or below the desk introducing partial occlusion. Moreover, some students talk in a room with another person asking for help where both are visible in the webcam, while others might speak with another person who is not visible in any of the two cameras. Some speak with a low voice (i.e., whispering) while others speak normally. Many other variations are also present in this dataset, since we did not constrain the subjects in how to cheat.

Note that the SVM cheat classifier combines cheat type 2 and 4 into one class (i.e., Class 2), since both types involve speech. Moreover, cheat type 3 is not detected by the SVM

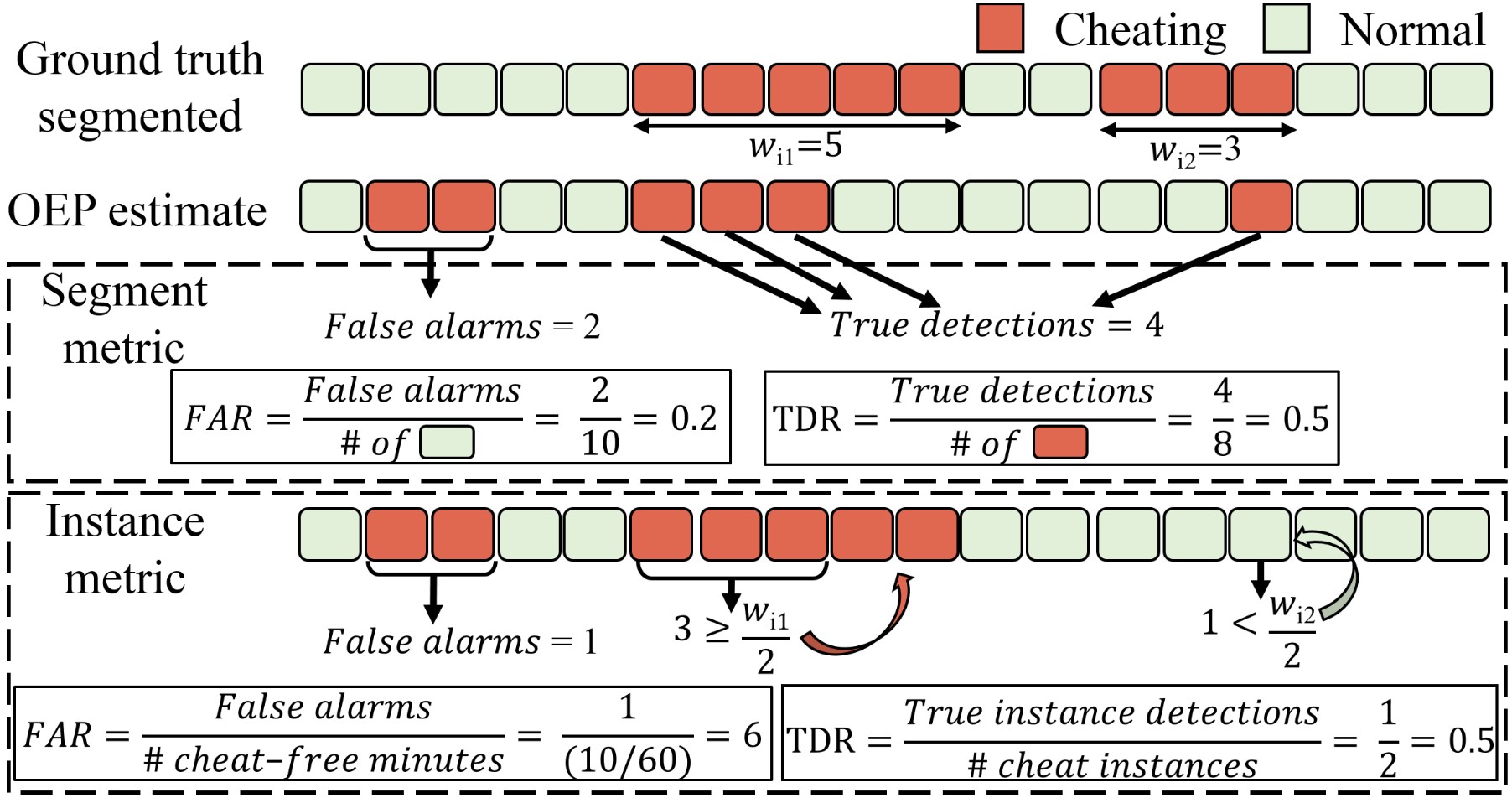


Fig. 10: An example of segment and instance-based metrics.

cheat classifier; instead, we detect it by the active window detection module which delivers an immediate cheat decision as seen in Fig. 2.

## V. EXPERIMENTAL RESULTS

In this section, we design experiments to answer the following questions: 1) How well can the system detect cheating? 2) How do different feature sets affect the performance? 3) What is the detectability of each cheat type? 4) Is there any correlation between the six components of the OEP system? 5) What is the system efficiency at a component and system level? We now discuss different aspects of our experiments. We start by explaining the evaluation procedure. Then we analyze the individual performance for a couple of basic components of our OEP system. After that, we test the performance of the entire OEP system. Finally, we describe the OEP system efficiency.

### A. Performance Evaluation

We define two metrics to evaluate the OEP system, a segment-based metric and an instance-based metric, with an example in Fig. 10. The segment-based metric evaluates the estimated classifier decisions at the segment level, which is the most straightforward measurement of the classification accuracy. A cheating instance is defined for the entire duration of one continuous cheating behavior, regardless of how long it is. The instance-based metric evaluates the detection accuracy based on the unit of cheating instance. Therefore, it is the “perceived” system accuracy of the user, and can answer questions such as “if a test taker cheats 10 times, how many times can OEP detect?” Both segment- and instance-based metrics are represented by True Detection Rate (TDR) and False Alarm Rate (FAR), but computed in different ways.

*a) Segment-based metric:* For segment-based metric, TDR is calculated by:

P*i* # detected cheating segments of subject *i*

TDR = P*i* # groundtruth cheating segments of subject *i,*

(4)

where *i* denotes the test subject ID. Since it is also important to not claim that a test taker is cheating when he/she is not, we compute FAR by:

P*i* # of false cheat segments of subject *i*

|  |  |  |  |
| --- | --- | --- | --- |
| Kernel | Dim= 50 | Dim= 100 | Dim= 200 |
| linear | 86.85% | 85.63% | 88.09% |
| quadratic polynomial | 73.55% | 73.61% | 74.53% |
| cubic polynomial | 78.61% | 81.91% | 74.50% |
| radial basis function | 93.38% | 93.43% | 94.25% |
| sigmoid | 81.51% | 83.94% | 82.74% |

TABLE II: Accuracy of classifying the validation data using SVM with different kernel functions and PCA dimensions.

|  |  |  |  |
| --- | --- | --- | --- |
| *γ* | Dim= 50 | Dim= 100 | Dim= 200 |
| 0.1 | 83.94% | 85.21% | 84.73% |
| 1 | 92.63% | 93.82% | 93.02% |
| 5 | 94.23% | 92.18% | 93.38% |
| 10 | 88.90% | 90.12% | 88.88% |

TABLE III: Accuracy of classifying the validation data using RBF kernel with different *γ* and PCA dimensions.

*b) Instance-based metric:* As illustrated in Fig. 10, to compute the instance-based metric, we filter the segment-based classification results in the following way. If more than 50% of the segments, regardless of their relative locations, within a cheating instance are correctly classified as cheating, this is a correctly detected instance. Otherwise, it is a miss detection at the instance level. The TDR in the instance-based metric is defined as:

P # detected cheating instances of subject *i i*

TDR = P # cheating instances of subject *i .* (6) *i*

To evaluate false alarm in the instance-based metric, as long as the number of consecutively detected false cheat segments is over *sf*, we define this as a falsely detected instance, regardless of its length. Since the instances within the cheat-free portion of the session is not well defined, we compute FAR w.r.t. the total length (in minutes) of cheat-free videos. Finally, the FAR in the instance-based metric is defined as,

P*i* # of false cheat instances of subject *i*

[[2]](#footnote-2) = P*i* # of cheat-free minutes of subject *i .* (7)

### B. Basic Component Analysis

In this section we demonstrate the accuracy of the two individual components, text and speech detection, which are the most important ones among all six components. The other components are evaluated along with the entire OEP system in the remaining sections. First of all, we set the parameters used in the six basic components as the following: *τ*0 = 3, *τv* = 0*.*9, *k* = 1, *d* = 0*.*6 meters, , *τs* = 5*,*000 and *sm* = 50. All experimental results reported in this section are evaluated with a 5-fold cross-validation on the positive and negative training samples as seen in Fig. 4

for text, and Table I for speech.

1. *Text detection analysis:* In text detection, the key parameters are the PCA dimensionality and the type of SVM kernel. Different choices of the parameters will affect the text detection performance. Using a two-class SVM [5], Table II illustrates the detection performance on the validation dataset, with different PCA dimensions and types of SVM kernel. Note

|  |  |  |  |
| --- | --- | --- | --- |
| *Ls* | 0.5s | 1s | 2s |
| Accuracy | 95.99% | 98.14% | 99.72% |

TABLE IV: Accuracy of classifying audio samples with different *Ls* lengths.

|  |  |
| --- | --- |
| Kernel | Accuracy |
| linear | 94.37% |
| quadratic polynomial | 95.68% |
| cubic polynomial | 95.99% |
| raidal basis function | 60.65% |
| sigmoid | 68.17% |

TABLE V: Accuracy of classifying audio samples using SVM with different kernel functions.

that reducing the dimensionality does not significantly reduce the detection performance. From this table, we see that the radial basis function (RBF) performs better than other kernels. Since the RBF kernel relies on a good choice of *γ*, we tested the detection performance using RBF kernel with different *γ* values as seen in Table III. It appears that using the SVM with RBF kernel (*γ* = 5) performs best on the validation dataset, where the feature dimension has been reduced to 50. We use these specific parameters in our final OEP system.

1. *Speech detection analysis:* We first analyze the speech detection performance with different acoustic segment lengths *Ls*. The testing results in Table IV illustrates that the larger the segment size, the higher accuracy can be achieved. The reason is that the longer audio segment carries more information about speech. However, in a real-world situation the longer the segments are, the more likely the short speech instances will miss detection. To balance between these two cases, we choose the fixed duration *Ls* as 500ms with a 100ms shift.

In order to choose the best kernel, we train the SVM classifiers using different kernels, and Table V gives the testing accuracy. From this table, we can see that the cubic polynomial function performs best over other kernels. Moreover, we test the performance of SVM using cubic polynomial kernel with different *γ* values, and it appears *γ* = 0*.*0072 generates the highest accuracy on the testing sound samples.

### C. OEP System Analysis

1. *Experimental setup:* All experiments are based on partitioning the dataset into two equal folds in the subject space for training and testing, while keeping the numbers of real and acting test takers equal between the two folds. This partition is repeated in three trials while maintaining the distribution of real vs. acting subjects. All reported results in the remaining section are based on the average of three trials. We set the window size *s* to 5 seconds, and the window shifts with an overlap of 1 second, which corresponds to an 80% overlap between consecutive segments. We set *sf* to 3 for computing the FAR in the instance-based metric. The multiclass SVM of the cheat classifier uses a linear kernel with the cost set to 10 [5], and all other parameters are set to the default values by LIBSVM.
2. *Feature analysis:* We start by analyzing the characteristics of the covariance features in the context of cheat



0.65

0.33

0.00

0.42

0.67

0.33

0.92

0.00

0.58

0.05

0.00

0.00

0.67

0.75

1.00

0.42

0.58

0.75

0.00

0.17

0.67

0.05

1.00

0.17

0.00

facePSR

textProb

voiceProb

gazeLR

gazeUD

facePSR

textProb

voiceProb

gazeLR

gazeUD

Fig. 11: Comparing the importance of different correlation of the five OEP components.

classification performance. First of all, we attempt to compare two different methods for computing the covariance features. The first method is to compute C as in Section III-H1. In the second method, we compute the 3 × 3 covariance matrix from each OEP component independently, and extract a 6dimensional covariance feature due to symmetry. Concatenating that of all five components (i.e., excluding v*ph*) results with a 30-dimensional covariance feature C¯. The difference between C and C¯, is that C has the ability to highlight, if any, the correlation across the five OEP components, whereas C¯ only finds the correlation within the statistics of each component. When comparing two types of covariance features in cheat classification, we observe that the first one, C, achieves higher classification accuracy, which indicates that incorporating cross-component correlation in the high-level feature benefits cheat classification.

The covariance feature C has a total of 120 dimensions. Within this large feature pool, which individual features are most relevant (or important) to the cheat classification task? To answer this question, we apply an AdaBoost feature selection technique [33] to select the most discriminative features among all elements of C. Given the training data in each of three trials, Adaboost selects the top 40 features from the 120 features of C. By repeating this for all three trials, we count how many times a feature has been selected, and normalize the counts by subtracting the minimum count and dividing with the difference of the maximum and minimum counts. This leads to the importance of correlation map in Fig. 11.

Some important observations can be made: (1) The voice detection component has a significant role in detecting cheat behaviors when combined with the gaze estimation. This means when a test taker cheats by asking a friend in the room, or by talking on the phone, he/she tends to change his head gaze direction. The same applies to the text detection and gaze components. (2) The inner-correlation of the component is observed as seen on the diagonal of Fig. 11, where the text, speech, and PSR vectors have high importance in the OEP system. The importance of the face PSR component is also relatively high, which is understandable, because for a test taker to cheat, this normally requires him to stop looking directly to the screen (i.e., webcam), and hence the PSR value changes accordingly. (3) A large number of correlation across components tend to have no importance, and therefore do not provide useful information in detecting cheat behaviors, such as facePSR & voiceProb and voiceProb & textProb.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| −0.01 0 0.01 0.02 0.03 0.04 0.05 0.06 −0.01 0 0.01 0.02 0.03 0.04 0.05 0.06 −0.1 0 0.1 0.2 0.3 0.4 0.5 0.6  FAR FAR Instance−based FAR (cheat/minute)  Fig. 12: Segment-based performance compar- Fig. 13: Performance comparison when using Fig. 14: Instance-based performance evaluaison via Binary SVM vs. Multi-class SVM. various window sizes. tion.   |  |  |  | | --- | --- | --- | | Cheating type | Detection rate | FAR distribution | | Text detection | 85.8% | 50.8% | | Speech detection | 89.3% | 43.9% | | Phone detection | 100.0% | 5.3% | |

TABLE VI: Error analysis of the OEP system at 2% FAR.

*c) OEP system results:* In our cheat detection classifier, we observe that using a multi-class SVM achieves a higher performance compared to the two-class SVM, as shown in Fig 12. This is partly because the positive class (i.e., all cheating behaviors) contains extremely diverse types of cheating, varying from reading text to verbally asking through speech events, which implies huge variations in the feature space. Hence, it is challenging to find a single hyperplane to best discriminate the negative class from the positive class. In contrast, using a four-class SVM defines multiple hyperplanes to better separate the four classes locally, which results in better overall cheating vs. non-cheating classification.

We further explore the temporal segments by changing the window size *s* as shown in Fig. 13. Here *s* is assigned to be 3, 5, or 7 seconds. We avoid selecting *s >* 7 because the majority of cheat behaviors tend to be short in duration. Note that the best performance is achieved when *s* = 5 seconds, with a TDR of 0*.*87 ± 0*.*03 at an FAR of 0*.*02.

Using the experimental setup based on the best parameters, we evaluate our system using the instance-based metric. The result is illustrated in Fig. 14. We see that our OEP system is able to detect cheating at an instance-based TDR of 0*.*80±0*.*04 and an FAR of 0*.*2 cheats per minute. This means that on average only one false alarm occurs per five minutes of the normal cheat-free exam.

Given the best results in the segment-based metric of Fig. 13, we are interested in what types of cheating behavior constitute the missing detection error and false alarm error. Based on the ground truth labels of the segments, we can categorize each wrongly classified segment (either miss detected one or false alarm one) into one of the three cheat classes, and illustrate the results in Table VI. We realize that speech detection performs better than text detection at a TDR equal to 89*.*3% with an FAR set to 2%. This is expected: detecting text from the wearcam is very challenging due to resolution, lighting and perspective distortion. It is also found that the number of false alarms related to text is also higher than speech. The phone detection has shown to work accurately for

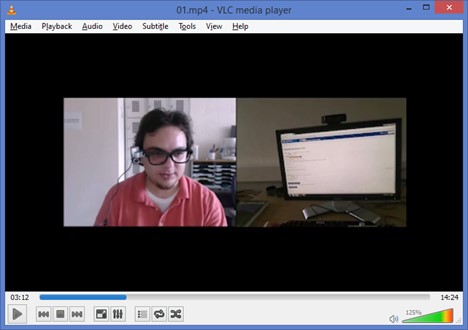


Fig. 17: The GUI for human proctoring used by three proctors.

detecting the cheat instances when test takers use the phone. Part of the reason is the limited phone-based cheat samples in the database - only 4% of cheat instances (23 cases for training and testing) as seen from Fig. 9. On the other hand, introducing the phone detection module to the system is accompanied with false alarms equal to 5*.*3% of all FAR. We show the entire classification results of two subjects in Fig. 15 and some of the system failure cases in Fig. 16.

### D. Performance of Human Proctoring

Human proctoring is the most common approach of validating online exams nowadays, by monitoring the test taker visually and acoustically via a webcam. In order to access its performance and contrast with our OEP system, we conduct an experiment imitating a human proctoring system, similar to the services offered by ProctorU. All testing videos used in our system were provided to three different people with experience in teaching, along with a graphical user interface (GUI) designed to manually record the cheating instances, as shown in Fig. 17. The GUI contained only one button which toggles between *Cheat* and *Stopped cheating* when clicked. The proctor had the ability to run one or two videos at the same time to imitate a real proctoring environment where one proctor usually “watches” multiple tests simultaneously. The proctors were not given any instructions other than to click the cheat button at the beginning of a cheat behavior, and to click again at the end of that same behavior.

After collecting the results of the three proctors, we compare them with the results of the OEP system individually, as well as jointly in two different schemes: (a) the majority of the proctors decision (i.e., two out of three need to agree), and (b) the intersection of the proctors decision (i.e., all three need to agree). Table VII shows the total cheat time labeled

|  |
| --- |
| Fig. 15: Results of Subject 10 (top) and Subject 16 (bottom) based on the chosen threshold that produces a segment-based FAR of 0*.*2. Subject 10 cheats 15 times during the exam, while 2 of them are not detected. Subject 16 cheats 10 times, where 3 of them are not detected along with 1 false alarm. Best viewed in color.    Fig. 16: Failure examples of the OEP system, showing the frames from both of the webcam and wearcam along with the estimated cheat behavior probability for a specific duration in the test taking illustrated by the x-axis. (a, b, c) represent cases where the OEP struggles to recognize the cheat activity of type 1, 2, and 4, respectively. (d, e, f) are false alarms where the system claims the subjects are cheating, but |

the ground truth reflects otherwise.

by the proctors, the segment TDR and FAR, and the instance TDR and FAR. Based on the ground truth labeling, the testing videos contain cheating behaviors for a total time of 3*,*199 seconds. It is clear that the human proctors reported cheating durations much larger than the actual cheat time, which is reflected negatively in the FAR measurements. Part of the reason is the slow reaction of humans towards switching on/off the cheat duration. Typically, ∼2 seconds are needed before confirming that the student has started/ended the cheating behavior. Furthermore, human proctors lose their attention span in some parts of the proctoring session, which leads to lower TDR. Note that the OEP results are chosen at an operation point where the TDR is the most similar to the human performance of “Majority”, which appears to be the best among all human performance. In general, when achieving the same TDR, the OEP system can maintain a lower FAR than the human proctors. We recognize that, in this comparison, the precise onset and offset locations of a cheating duration matter, which may not be the case in a real-world scenario and that would change the comparison accordingly.

Segment metric Instance metric

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Results of | Cheat time (s) | TDR | FAR | TDR | FAR |
| Proctor 1 | 4,567 | 0.86 | 0.13 | 0.88 | 0.90 |
| Proctor 2 | 4,728 | 0.85 | 0.14 | 0.83 | 0.69 |
| Proctor 3 | 5,504 | 0.71 | 0.22 | 0.72 | 0.77 |
| Majority | 4,650 | 0.87 | 0.13 | 0.85 | 0.77 |
| Intersection | 2,758 | 0.60 | 0.06 | 0.58 | 0.33 |
| OEP | 2,958 | 0.87 | 0.02 | 0.85 | 0.42 |

TABLE VII: Comparison of human proctoring and OEP system.

### E. System Efficiency

The six OEP basic components are all implemented in C++. The high-level feature extraction and cheat classification are implemented in Matlab. Table VIII shows the system efficiency break down in frame per second (FPS), while the system runs on a personal desktop computer with Windows 8 (Intel i5 CPU at 3*.*0 GHz with 8 GB RAM). It can be observed that the computation cost of Stage 2 cheat detection is negligible compared to that of the basic components. Among the six basic components, text detection is the slowest one which requires

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Stage 1- Basic component | FPS | | User Verification | 10 | | Text detection | 4 | | Speech detection | 25 | | Window detection | 1,000 | | Gaze estimation | 175 | | Phone detection | 37 | | |  |  | | --- | --- | | Stage 2- Cheat detection per seg. | FPS | | Features extraction | 1,816 | | Cheat classification | 932 | |

TABLE VIII: Efficiency of basic components, feature extraction and classification of the OEP system.

238 ms per frame. Based on these costs, if a test taker takes an exam for 1 minute, our OEP system would require a total of ∼6 minutes to finish processing the videos from two cameras along with the audio. Note that this 6X slower-than-real-time speed is based on the assumption that all six basic components process every frame in 25 FPS videos. In reality, it is very likely that we may process at a lower frame rate, yet still maintain similar detection performance, since for example, the test taker would need *a few* seconds in text-based cheating.

## VI. DISCUSSION

The main contribution of this work is to present a comprehensive framework for online exam proctoring. While we have achieved good performance in our evaluation, our framework can certainly be improved in a number of ways. For the basic components, we can either apply more advanced algorithms for each component, such as the deep learning-based feature representation, typing-based continuous authentication [25], [26], face alignment-based pose estimation [12], [18], [19], upper body alignment [20], and model personalization [6]. We may also expand the array of basic components, to include additional components such as pen detection. For cheat classification, we can explore temporal-spatial dynamic features, similar to the work in video-based activity recognition [36]. Moreover, the system efficiency can also be improved while maintaining a high accuracy in recognizing cheat events as suggested in [11], by selecting more suitable features and classifiers, as well as selecting a smaller number of frames instead of utilizing all frames.

We recognize that there always exists a possibility that concealed cheating activities might happen outside the fields of view of both cameras. To remedy this, our system plans to generate random commands, such as asking the test taker to look around or under the desk to check the surrounding environment of exam. To detect whether the test taker has tampered with the sensors, once in a while our system can display a simple icon on the computer screen to validate that the wearcam can “see” it, or play a quick sound clip to validate that the microphone can “hear” it. The randomness of such commands and intervention will likely make our system more robust against deliberate cheating behavior.

Note that the definition of cheating behavior depends on the *context of the exam*, such as oral exam, open-book exam, etc. Our proposed hybrid two-stage algorithm enables the user to take into consideration such context of the exam. The six basic components extracted in the first stage can be considered as system building blocks, which are reconfigurable based on the context of the exam and the test taker’s preference. For example, if the exam is an open-book exam, the OEP system should exclude the text detection component. Some other types of exam might require the test taker to talk such as oral exams, and hence removing the speech detection component is necessary.

Even with all the aforementioned system enhancements, it is possible that the automatic OEP system might not achieve perfect performance (i.e., detecting all cheating behaviors with no false alarm). We note that even in traditional classroom proctoring, it is likely that the proctor will fail to detect some cheating behaviors, due to either the attention span of the proctor or highly concealed action. Therefore, as long as OEP can capture the majority of cheating behaviors with reasonably small false alarm, it will be a useful contribution to online education. Furthermore, we may also allow humans to manually inspect the instances with high probability of cheating from our system. For example, setting a proper threshold in Fig. 15 detects all such instances. This manual inspection helps to verify the true detections, as well as suppress the false alarms. Hence, the combination of using OEP to detect likely cheating instances within the entire session, and the manual inspection on a very small subset of data, can achieve an excellent trade-off between system accuracy and cost. Finally, as visual analysis technology progresses, it is obvious that the workload of manual inspection will become less and less.

## VII. CONCLUSIONS

This paper presents a multimedia analytics system for online exam proctoring, which aims to maintain academic integrity in e-learning. The system is affordable and convenient to use from the text taker’s perspective, since it only requires having two inexpensive cameras and a microphone. With the captured videos and audio, we extract low-level features from six basic components: user verification, text detection, speech detection, active window detection, gaze estimation, and phone detection. These features are then processed in a temporal window to acquire high-level features, and then are used for cheat detection. Finally, with the collected database of 24 test takers representing real-world behaviors in online exam, we demonstrate the capabilities of the system, with nearly 87% segment-based detection rate across all types of cheating behaviors at a fixed FAR of 2%. These promising results warrant further research on this important behavior recognition problem and its educational application.

## VIII. ACKNOWLEDGEMENTS

This project was sponsored in part by the Michigan State University Targeted Support Grants for Technology Development (TSGTD) program. The authors thank anonymous volunteers for participating in our OEP database collection. The authors also thank Baraa Abu Dalu, Muath Nairat, and Mohammad Alrwashdeh for contributing to the study of human proctoring.

## REFERENCES

1. I. E. Allen and J. Seaman. Grade change: Tracking online education in the united states, 2013. *Babson Survey Research Group and Quahog Research Group, LLC. Retrieved on*, 3(5), 2014.
2. Y. Atoum, S. Srivastava, and X. Liu. Automatic feeding control for dense aquaculture fish tanks. *IEEE Signal Processing Letters*, 22(8):1089– 1093, 2015.
3. D. L. Baggio. Enhanced human computer interface through webcam image processing library. *Natural User Interface Group Summer of Code Application*, pages 1–10, 2008.
4. S. V. Bailey and S. V. Rice. A web search engine for sound effects. In *Audio Eng. Society Convention 119*. Audio Eng. Society, 2005.
5. C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM T-TIST*, 2(3):27, 2011.
6. J. Chen and X. Liu. Transfer learning with one-class data. *Pattern Recognition Letters*, 37:32–40, 2014.
7. G. Cluskey Jr, C. R. Ehlen, and M. H. Raiborn. Thwarting online exam cheating without proctor supervision. *Journal of Academic and Business Ethics*, 4:1–7, 2011.
8. P. Guo, H. feng yu, and Q. Yao. The research and application of online examination and monitoring system. In *IT in Medicine and Education, 2008. IEEE Int. Sym. on*, pages 497–502, 2008.
9. D. Hoiem, Y. Ke, and R. Sukthankar. Solar: sound object localization and retrieval in complex audio environments. In *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, volume 5, pages

429–432, 2005.

1. H. Hung and D. Gatica-Perez. Estimating cohesion in small groups using audio-visual nonverbal behavior. *IEEE Trans. Multimedia*, 12(6):563– 575, 2010.
2. Y.-G. Jiang, Q. Dai, T. Mei, Y. Rui, and S.-F. Chang. Super fast event recognition in internet videos. *IEEE Trans. Multimedia*, 17(8):1174– 1186, 2015.
3. A. Jourabloo and X. Liu. Pose-invariant 3d face alignment. In *Proc. Int. Conf. Computer Vision (ICCV)*, pages 3694–3702, 2015.
4. I. Jung and H. Yeom. Enhanced security for online exams using group cryptography. *Education, IEEE Trans. on*, 52(3):340–349, 2009.
5. V. Kilic, M. Barnard, W. Wang, and J. Kittler. Audio assisted robust visual tracking with adaptive particle filtering. *IEEE Trans. Multimedia*, 17(2):186–200, 2015.
6. D. L. King and C. J. Case. E-cheating: Incidence and trends among college students. *Issues in Information Systems*, 15(1), 2014.
7. I. Lefter, L. J. Rothkrantz, and G. J. Burghouts. A comparative study on automatic audio–visual fusion for aggression detection using metainformation. *Pattern Recognition*, 34(15):1953–1963, 2013.
8. X. Li, K.-m. Chang, Y. Yuan, and A. Hauptmann. Massive open online proctor: Protecting the credibility of moocs certificates. In *ACM CSCW*, pages 1129–1137. ACM, 2015.
9. X. Liu. Video-based face model fitting using adaptive active appearance model. *Image and Vision Computing*, 28(7):1162–1172, July 2010.
10. X. Liu, N. Krahnstoever, T. Yu, and P. Tu. What are customers looking at? In *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance (AVSS)*, pages 405–410, London, UK, Sept. 2007.
11. X. Liu, T. Yu, T. Sebastian, and P. Tu. Boosted deformable model for human body alignment. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Anchorage, Alaska, June 2008. IEEE.
12. L. Nguyen, D. Frauendorfer, M. Mast, and D. Gatica-Perez. Hire me: Computational inference of hirability in employment interviews based on nonverbal behavior. *IEEE Trans. Multimedia*, 16(4):1018–1031, 2014.
13. A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int. J. Comput. Vision*, 42(3):145– 175, 2001.
14. M. Reale, S. Canavan, L. Yin, K. Hu, and T. Hung. A multi-gesture interaction system using a 3-d iris disk model for gaze estimation and an active appearance model for 3-d hand pointing. *IEEE Trans. Multimedia*, 13(3):474–486, 2011.
15. W. Rosen and M. Carr. An autonomous articulating desktop robot for proctoring remote online examinations. In *Frontiers in Education Conf., 2013 IEEE*, pages 1935–1939, 2013.
16. J. Roth, X. Liu, and D. Metaxas. On continuous user authentication via typing behavior. *IEEE Trans. Image Process.*, 10:4611–4624, Oct. 2014.
17. J. Roth, X. Liu, A. Ross, and D. Metaxas. Investigating the discriminative power of keystroke sound. *IEEE Trans. Inf. Forens. Security*, 10(2):333–345, 2015.
18. D. Sadlier and N. O’Connor. Event detection in field sports video using audio-visual features and a support vector machine. *IEEE Trans. Circuits Sys. Video Technol.*, 15(10):1225–1233, 2005.
19. M. Savvides, B. V. Kumar, and P. Khosla. Face verification using correlation filters. *3rd IEEE Automatic Identification Advanced Technologies*, pages 56–61, 2002.
20. J. Shi and C. Tomasi. Good features to track. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 593–600,

1994.

1. E. Spriggs, F. De la Torre, and M. Hebert. Temporal segmentation and activity classification from first-person sensing. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages

17–24, 2009.

1. A. Tsukada, M. Shino, M. Devyver, and T. Kanade. Illumination-free gaze estimation method for first-person vision wearable device. In *Proc. Int. Conf. Computer Vision (ICCV) Workshops*, pages 2084–2091, 2011.
2. O. Tuzel, F. Porikli, and P. Meer. Pedestrian detection via classification on riemannian manifolds. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(10):1713–1727, 2008.
3. P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 511–518, 2001.
4. A. Wahid, Y. Sengoku, and M. Mambo. Toward constructing a secure online examination system. In *Proc. of the 9th Int. Conf. on Ubiquitous Information Management and Communication*, page 95. ACM, 2015.
5. B. Xiao, P. Georgiou, B. Baucom, and S. Narayanan. Head motion modeling for human behavior analysis in dyadic interaction. *IEEE Trans. Multimedia*, 17(7):1107–1119, 2015.
6. Y. Zhang, X. Liu, M.-C. Chang, W. Ge, and T. Chen. Spatio-temporal phrases for activity recognition. In *Proc. European Conf. Computer Vision (ECCV)*, pages 707–721, Florence, Italy, Oct. 2012.

Yousef Atoum is a Ph.D. student in the Department of Electrical and Computer Engineering at Michigan State University. He received a B.S. degree in Computer Engineering from Yarmouk University, Jordan, in 2009, and a M.S. degree in Electrical and Computer Engineering from Western Michigan University, in 2012. His research interests include object tracking, computer vision and pattern recognition.

Liping Chen is a Ph.D. student in the Department of Mathematics and a M.S. student in the Department of Computer Science and Engineering at Michigan State University. He received his B.S. degree in

Information and Computational Mathematics from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2006. His research interests include numerical analysis, tensor eigenvalue, and computer vision.

Alex X. Liu received his Ph.D. degree in Computer Science from The University of Texas at Austin in 2006. He received the IEEE & IFIP William C. Carter Award in 2004, a National Science Foundation CAREER award in 2009, and the Michigan State University Withrow Distinguished Scholar Award in 2011. He is an Associate Editor of IEEE/ACM Transactions on Networking, an Associate Editor of IEEE Transactions on Dependable and Secure Computing, and an Area Editor of

Computer Communications. He received Best Paper Awards from ICNP-2012, SRDS-2012, and LISA-2010. His research interests focus on networking and security.

Stephen D. H. Hsu is Vice President for Research and Graduate Studies and Professor of Theoretical Physics at Michigan State University. Hsu graduated from the California Institute of Technology and received his doctorate from the University of California, Berkeley. A Junior Fellow of the Harvard Society of Fellows, he was on the faculty at Yale University before moving to the University of Oregon where he served as Director of the Institute for Theoretical Science. Hsus primary work has been in applications of quantum field theory, particularly

to problems in cosmology and fundamental physics. He has also made contributions in genomics and bioinformatics, finance, and in encryption and information security. He founded two Silicon Valley companiesSafeWeb and Robot Genius Inc., and is a scientific advisor to BGI Shenzhen.

Xiaoming Liu is an Assistant Professor at the Department of Computer Science and Engineering of Michigan State University. He received the Ph.D. degree in Electrical and Computer Engineering from Carnegie Mellon University in 2004. Before joining

MSU in Fall 2012, he was a research scientist at General Electric (GE) Global Research. His research interests include computer vision, patter recognition, biometrics and machine learning. As a co-author, he is a recipient of Best Industry Related Paper Award runner-up at ICPR 2014, Best Student Paper Award

at WACV 2012 and 2014, and Best Poster Award at BMVC 2015. He has been an Area Chair for numerous conferences, including FG, ICPR, WACV, ICIP, and CVPR. He is an Associate Editor of Neurocomputing journal. He has authored more than 100 scientific publications, and has filed 22 U.S. patents.

**Appendix 3**

**Tools Used**

**Software:**

1.MYSQL

2.Flask

3.OpenCV

4.Python IDE

5.DL Libraries

1. <http://cvlab.cse.msu.edu/project-OEP.html> [↑](#footnote-ref-1)
2. = P # of cheat-free segments of subject *i .* (5)

   *i* [↑](#footnote-ref-2)