ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/312676827

Automated Online Exam Proctoring

Article *in* IEEE Transactions on Multimedia · January 2017

DOI: 10.1109/TMM.2017.2656064

CITATIONS
237

READS
26,053

5 authors, including:

Xiaoming Liu
Michigan State University
282 PUBLICATIONS **26,623** CITATIONS

SEE PROFILE

All content following this page was uploaded by Xiaoming Liu on 17 December 2018.

The user has requested enhancement of the downloaded file.

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 higher-level 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

MASSIVE 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 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.

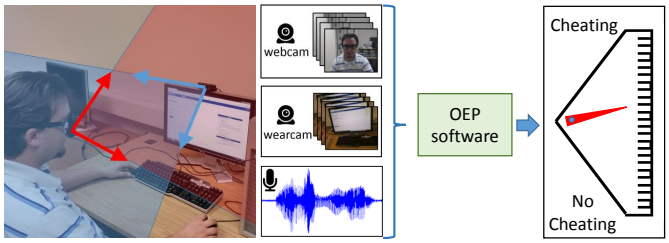

Fig. 1: Based on the audio-visual streams captured by a wearcam, a webcam with an integrated microphone, our OEP system *automatically* and *continuously* detects cheat behaviors during online exams.

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

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

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 middle-level 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].

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) semi-automated 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 right-hand 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 audio-video 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].

[1] http://cvlab.cse.msu.edu/project-OEP.html

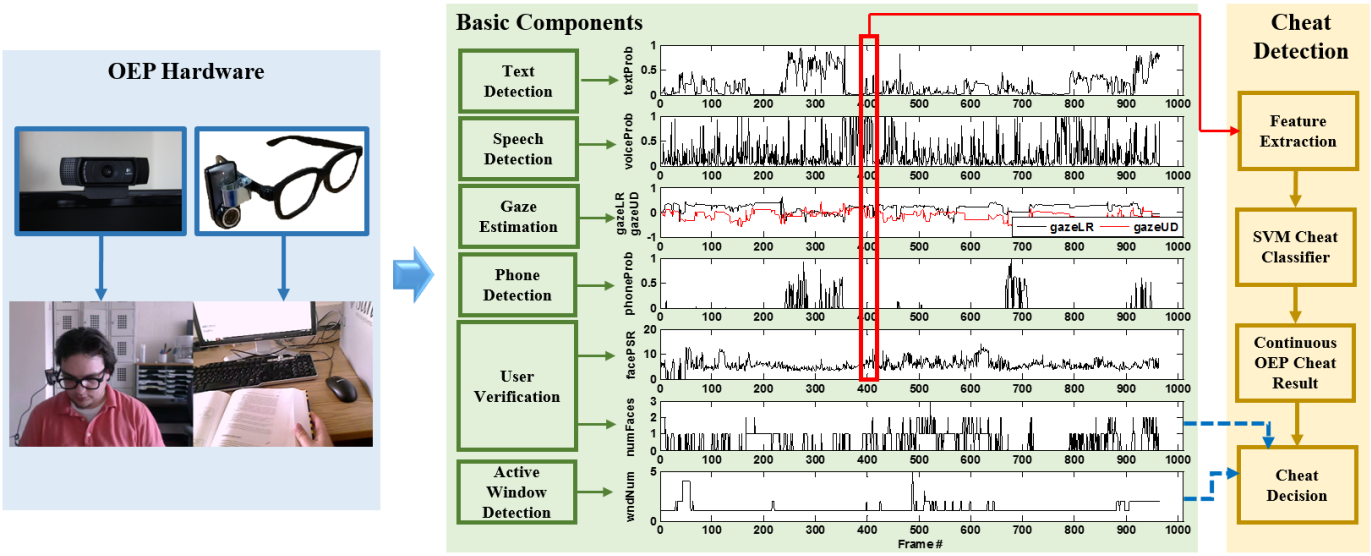

Fig. 2: The architecture of the Online Exam Proctoring (OEP) system.

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 high-level 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

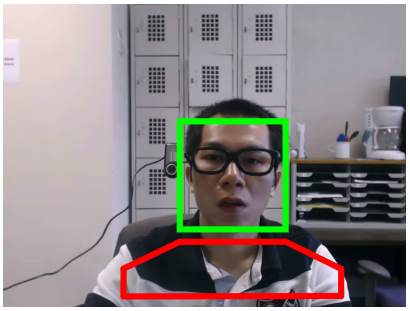

Fig. 3: The extracted region of the face (green) and the body (red).

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 $f_s = 25$ 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 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 $\mathbf{h}_f$. As shown in Fig. 3, from the body region of the images, we extract a 160-dim HSV color histogram of the clothing $\mathbf{h}_b$. 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 $\mathbf{I}_t$ is captured by the webcam, we first perform face detection. Depending on the number of detected faces $\mathbf{v}_n(t)$, we handle it correspondingly, as described in Algorithm 1.

Algorithm 1: User verification algorithm.

Data: A new frame $\mathbf{I}_t$, $\mathbf{h}_f$
Result: $\mathbf{v}_p$, $\mathbf{v}_n$
Initialization: $v = 0$, $c_0 = c_1 = 0$;
Viola-Jones face detector $\rightarrow \mathbf{v}_n(t)$;
switch $\boldsymbol{v}_n(t)$ **do**
  case 0
    if $c_0 > \tau_0$ **then**
      $p_t = \mathbf{v}_p(t) = c_0 = 0$; % warning is sent
    else
      $c_0 + +$;
      $\mathbf{v}_p(t) = \mathbf{v}_p(t-1)$;
      $p_t = F(\mathbf{v}_p(t), v, \bar{t}, \bar{p})$;
  case 1
    $c_0 = c_1 = 0$;
    if $v = 1$ **then**
      Compute $\mathbf{h}_t$, $p_b = \mathbf{h}_t^T \mathbf{h}_b$;
      if $p_b > \tau_v$ & $p_{t-1} > \tau_v$ **then**
        $p_t = F(\mathbf{v}_p(t), v, \bar{t}, \bar{p})$;
      else
        $v = 0$;
    if $v = 0$ **then**
      $\mathbf{c}_t = \mathbf{x}_t \bigotimes \mathbf{h}_f$, $\mathbf{v}_p(t) = \mathrm{PSR}(\mathbf{c}_t)$;
      $p_t = F(\mathbf{v}_p(t), v, \bar{t}, \bar{p})$;
      if $p_t > \tau_v$ **then**
        $v = 1$, $\bar{t} = t$, $\bar{p} = p_t$;
  case > 1
    if $c_1 > \tau_0$ **then**
      $p_t = \mathbf{v}_p(t) = c_1 = 0$; % warning is sent
    else
      $c_1 + +$;
      $\mathbf{v}_p(t) = \mathbf{v}_p(t-1)$;
      $p_t = F(\mathbf{v}_p(t), v, \bar{t}, \bar{p})$;

If only one face is detected in the new frame ($\mathbf{v}_n(t) = 1$); this is the most likely case since the test taker is required to take the exam alone. Let $\mathbf{x}_t$ be the appearance feature of the detected face, p_t 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 $\mathbf{c}_t = \mathbf{x}_t \bigotimes \mathbf{h}_f$, where $\mathbf{c}_t$ 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 $\mathbf{c}$ 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

Algorithm 2: Face verification probability F.

Data: $\mathbf{v}_p(t), v, \bar{t}, \bar{p}$
Result: p_t
if $v = 0$ **then**
  if $\boldsymbol{v}_p(t) < 5$ **then**
    $p_t = 0$;
  else if $\boldsymbol{v}_p(t) \geqslant 10$ **then**
    $p_t = 1$;
  else
    $p_t = \frac{1}{5}(\mathbf{v}_p(t) - 5)$;
else
  $p_t = \bar{p}e^{-k(t-\bar{t})}$;

an authenticated user. We denote the PSR value computed at time t as $\mathbf{v}_p(t)$, which is further converted into the probability measure of user authenticity p_t, by using the function F,

$$p_t = F(\mathbf{v}_p(t), v, \bar{t}, \bar{p}), \tag{1}$$

as explained in Algorithm 2. If p_t is larger than a predefined threshold τ_v, the face is verified, and we denote $\bar{t}$ the last verified time and $\bar{p}$ 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 $\mathbf{h}_t$, and compare it to $\mathbf{h}_b$. If their similarity p_b is larger than a threshold τ_v, p_t is calculated as $p_t = \bar{p}e^{-k(t-\bar{t})}$, where k is the decay speed of the exponential function. After $\Delta t = t - \bar{t}$ seconds from the last verification time, the face needs to be verified again even if $p_b > \tau_v$ 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 ($\mathbf{v}_n(t) = 0$). 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 $c_0 \leq \tau_0$, 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 ($\mathbf{v}_n(t) > 1$), we also consider some tolerance, similar to the case of $\mathbf{v}_n(t) = 0$. 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, $\mathbf{v}_n$, is a direct indication of cheating when $\mathbf{v}_n(t) \neq 1$. However, the facePSR alone, $\mathbf{v}_p$, 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.

$\mathbf{v}_t$, 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 audio-visual 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 L_s. Due to its simplicity and robustness, we decide to adopt the latter approach. It is a trade-off to determine the length of L_s, 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 $\mathbf{v}_v$, 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 $\mathbf{v}_w$, obtained from the operational system API. Most of the time, there should be only one active window, which is the online exam itself. If $\mathbf{v}_w(t) > 1$ 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.

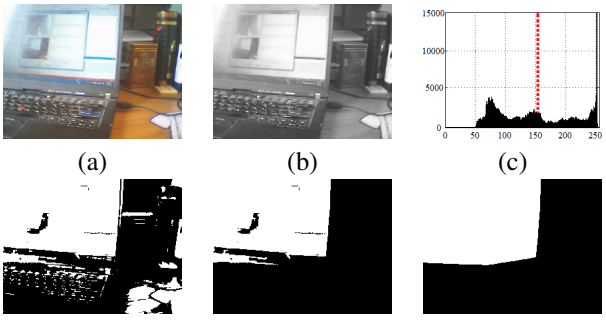

Fig. 5: Screen detection process: (a) input frame $\mathbf{I}_t$, (b) grayscale image, (c) histogram of (b), (d) converted binary image based on the threshold, (e) the largest region after connected component analysis, and (f) estimated screen using the convex hull of the largest region.

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 $\mathbf{c}_s$, width as w_s, and height as h_s. 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 $\hat{\mathbf{c}}_s$. We assume the distance between the test taker and the screen is set to a fixed distance of d. Knowing d, $\mathbf{c}_s$ and $\hat{\mathbf{c}}_s$, the head pose is calculated by

$$\mathbf{v}_g = \arctan \frac{\|\mathbf{c}_s - \hat{\mathbf{c}}_s\|}{d}. \tag{2}$$

It is obvious that we may only estimate $\mathbf{v}_g$ using the screen region when the screen is visible in the wearcam video. That is, when the head gaze is larger than θ_g, the screen is out-of-view 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 $\mathbf{v}_g > \theta_g$.

For each frame, we store the results of the gaze estimation into elements of two vectors, "gazeLR" and "gazeUD", which are denoted as $\mathbf{v}_{g1}$ and $\mathbf{v}_{g2}$, 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 $[-\frac{\pi}{2}, \frac{\pi}{2}]$, we normalize them such that $\mathbf{v}_{g1} \in [-1, 1]$, where -1 means the user is looking far left at an angle of $-\frac{\pi}{2}$ and 1 is towards the far right at $\frac{\pi}{2}$. The same applies to $\mathbf{v}_{g2}$.

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 τ_l and τ_s 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 $\mathbf{v}_{ph}$. Since the phone detection module detects phone with an area in the range of $[\tau_s, \tau_l]$, we normalize them such that $\mathbf{v}_{ph} \in [0, 1]$, representing the probability of detecting a phone in the frame. Since the vector

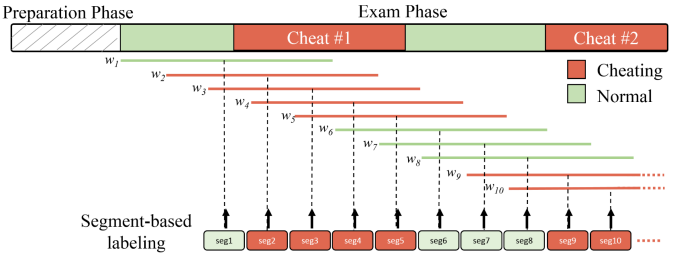

Fig. 6: Segment-based labeling process for a subject. In this example, the test taker cheats two times during the exam. Note how the window w shifts at exact increments with an 80% overlap. At each shift, a segment is formed and assigned a label based on the majority vote of ground truth labels that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

$\mathbf{v}_{ph}$ could be noisy, we apply a median filter of a fixed size s_m, to eliminate the random noise.

H. Cheating Behavior Detection

At this stage, we have the continuous output of the OEP basic components (i.e., $\mathbf{v}_p$,$\mathbf{v}_t$,$\mathbf{v}_v$,$\mathbf{v}_{g1}$,$\mathbf{v}_{g2}$,$\mathbf{v}_{ph}$,$\mathbf{v}_w$,$\mathbf{v}_n$), 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 $\mathbf{v}_w$ and $\mathbf{v}_n$ (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 middle-level 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 w_t, 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 $\mathbf{C}$.

The covariance feature is an effective visual feature used in many vision systems, including pedestrian detection [32]. Let $\mathbf{v}_i$ be the i^{th} component vector obtained from one segment. We compute a $sf_s \times 3$ matrix $\mathbf{A}_i = [\mathbf{v}_i \quad |\mathbf{v}_i'| \quad |\mathbf{v}_i''|]$, where $|\mathbf{v}_i'|, |\mathbf{v}_i''|$ are the absolute values of the first and second order derivatives, respectively. Due to the sparsity of $\mathbf{v}_{ph}$ (i.e., most elements of the vector are zeros as seen in Fig. 2), we exclude it from extracting covariance features. Therefore, combining $\mathbf{A}_i$ of all the remaining five vectors yields a $sf_s \times 15$ matrix, $\mathbf{A} = [\mathbf{A}_1\mathbf{A}_2...\mathbf{A}_5]$. To compute the covariance feature, we apply the following equation:

$$\mathbf{C} = \frac{1}{s-1}(\mathbf{A} - \text{mean}(\mathbf{A}))^T(\mathbf{A} - \text{mean}(\mathbf{A})), \qquad (3)$$

where mean() computes the mean across all rows. Since $\mathbf{C}$ 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.

2) 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.

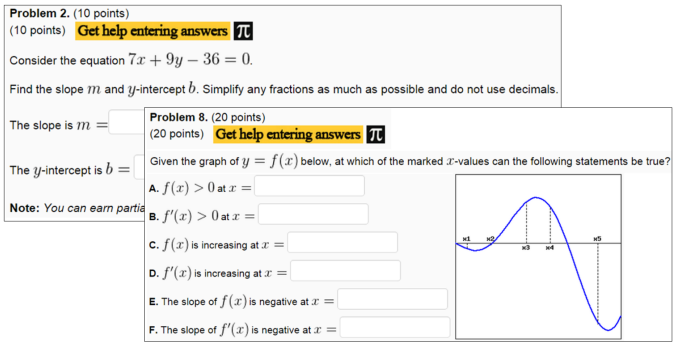

Fig. 7: Two questions of the mathematics exam that are given to test takers during data collection.

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 taking 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 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:

$$\text{TDR} = \frac{\sum_i \#\text{ detected cheating segments of subject } i}{\sum_i \#\text{ groundtruth cheating segments of subject } i}, \quad (4)$$

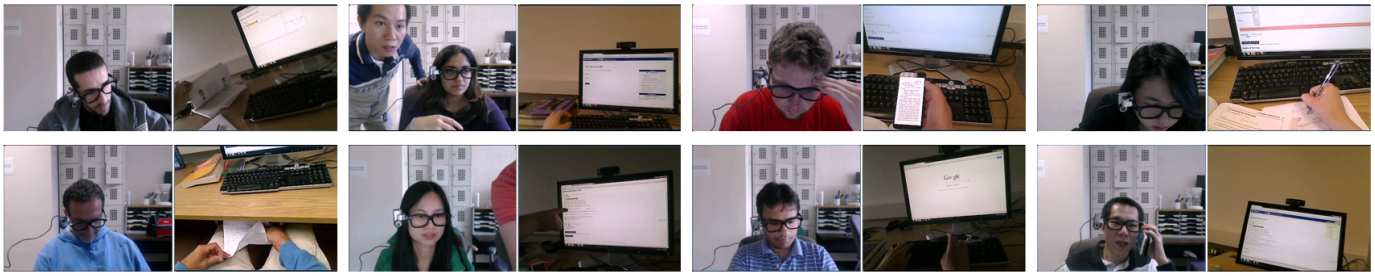

Fig. 8: OEP dataset examples illustrating various cheat types. The examples are grouped in pairs showing both webcam and wearcam at a specific time of the exam. The subjects are cheating from books, notes, papers, smartphones, the Internet, or asking someone in the room.

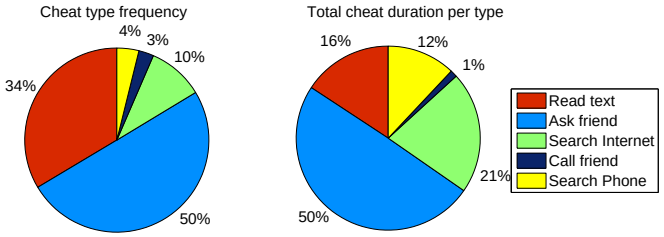

Fig. 9: Statistics of cheating behavior in the OEP dataset. Cheat types: (1) cheat from book/note/paper, (2) talk in room with a person, (3) use the Internet, (4) ask a friend over the phone, and (5) use a phone or other devices.

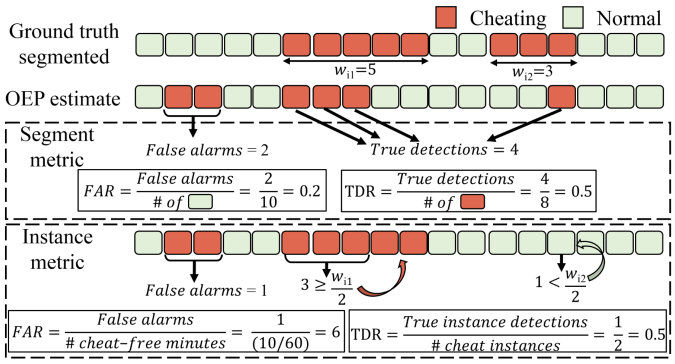

Fig. 10: An example of segment and instance-based metrics.

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:

$$\text{FAR} = \frac{\sum_i \#\text{ of false cheat segments of subject } i}{\sum_i \#\text{ of cheat-free segments of subject } i}. \quad (5)$$

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:

$$\text{TDR} = \frac{\sum_i \#\text{ detected cheating instances of subject } i}{\sum_i \#\text{ cheating instances of subject } i}. \quad (6)$$

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.

To evaluate false alarm in the instance-based metric, as long as the number of consecutively detected false cheat segments is over s_f, 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,

$$\text{FAR} = \frac{\sum_i \#\text{ of false cheat instances of subject } i}{\sum_i \#\text{ of cheat-free minutes of subject } i}. \quad (7)$$

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: $\tau_0 = 3$, $\tau_v = 0.9$, $k = 1$, $d = 0.6$ meters, $\theta_g = \frac{\pi}{4}$, $\tau_l = 15,000$, $\tau_s = 5,000$ and $s_m = 50$. 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

L_s	0.5s	1s	2s
Accuracy	95.99%	98.14%	99.72%

TABLE IV: Accuracy of classifying audio samples with different L_s 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.

illustrates the detection performance on the validation dataset, with different PCA dimensions and types of SVM kernel. Note 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 ($\gamma = 5$) 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 L_s. 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 L_s 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 $\gamma = 0.0072$ generates the highest accuracy on the testing sound samples.

C. OEP System Analysis

a) 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 s_f to 3 for computing the FAR in the instance-based metric. The multi-class 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.

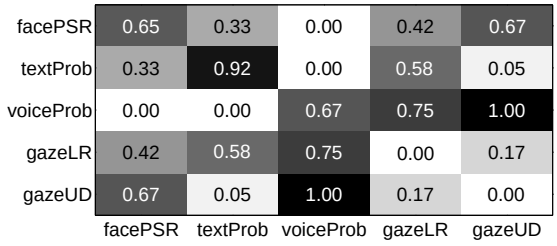

Fig. 11: Comparing the importance of different correlation of the five OEP components.

b) Feature analysis: We start by analyzing the characteristics of the covariance features in the context of cheat classification performance. First of all, we attempt to compare two different methods for computing the covariance features. The first method is to compute $\mathbf{C}$ as in Section III-H1. In the second method, we compute the 3×3 covariance matrix from each OEP component independently, and extract a 6-dimensional covariance feature due to symmetry. Concatenating that of all five components (i.e., excluding $\mathbf{v}_{ph}$) results with a 30-dimensional covariance feature $\bar{\mathbf{C}}$. The difference between $\mathbf{C}$ and $\bar{\mathbf{C}}$, is that $\mathbf{C}$ has the ability to highlight, if any, the correlation across the five OEP components, whereas $\bar{\mathbf{C}}$ 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, $\mathbf{C}$, achieves higher classification accuracy, which indicates that incorporating cross-component correlation in the high-level feature benefits cheat classification.

The covariance feature $\mathbf{C}$ 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 $\mathbf{C}$. Given the training data in each of three trials, Adaboost selects the top 40 features from the 120 features of $\mathbf{C}$. 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

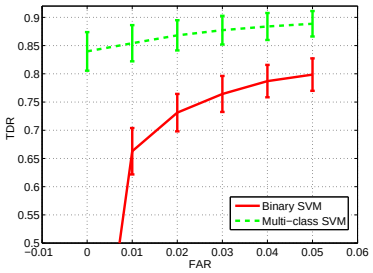

Fig. 12: Segment-based performance comparison via Binary SVM vs. Multi-class SVM.

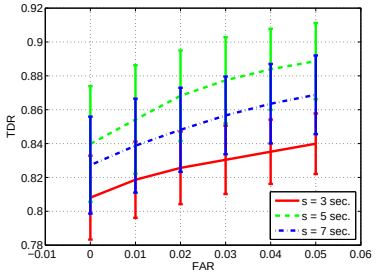

Fig. 13: Performance comparison when using various window sizes.

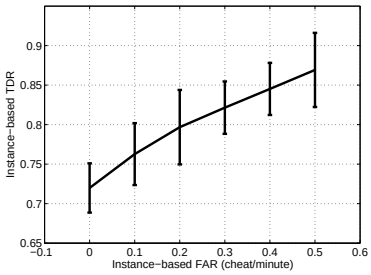

Fig. 14: Instance-based performance evaluation.

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.

provide useful information in detecting cheat behaviors, such as facePSR & voiceProb and voiceProb & textProb.

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 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 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, $\sim$2 seconds are needed before confirming that the student has started/ended the cheating

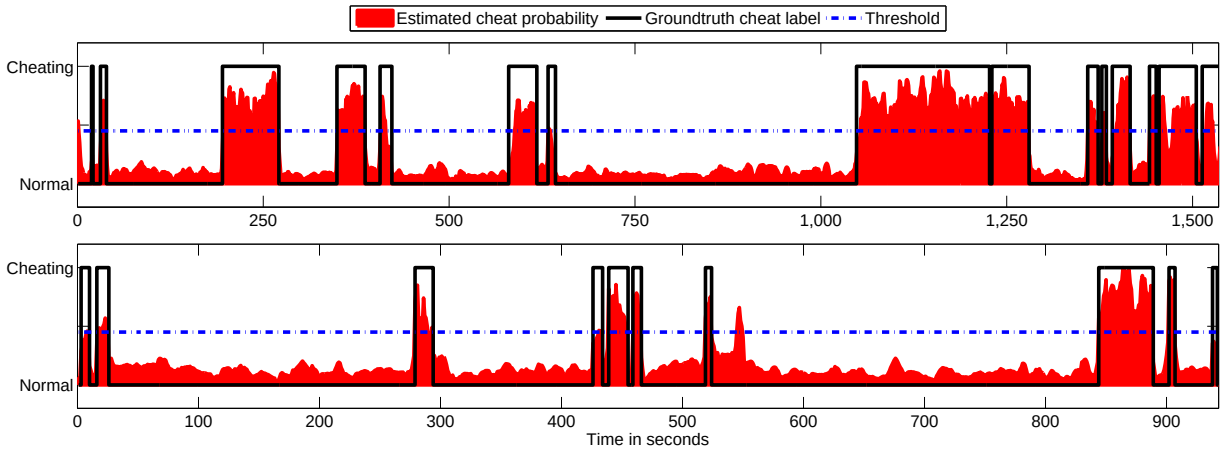

Fig. 15: Results of Subject 10 (top) and Subject 16 (bottom) based on the chosen threshold that produces a segment-based FAR of 0.2. Subject 10 cheats 15 times during the exam, while 2 of them are not detected. Subject 16 cheats 10 times, where 3 of them are not detected along with 1 false alarm. Best viewed in color.

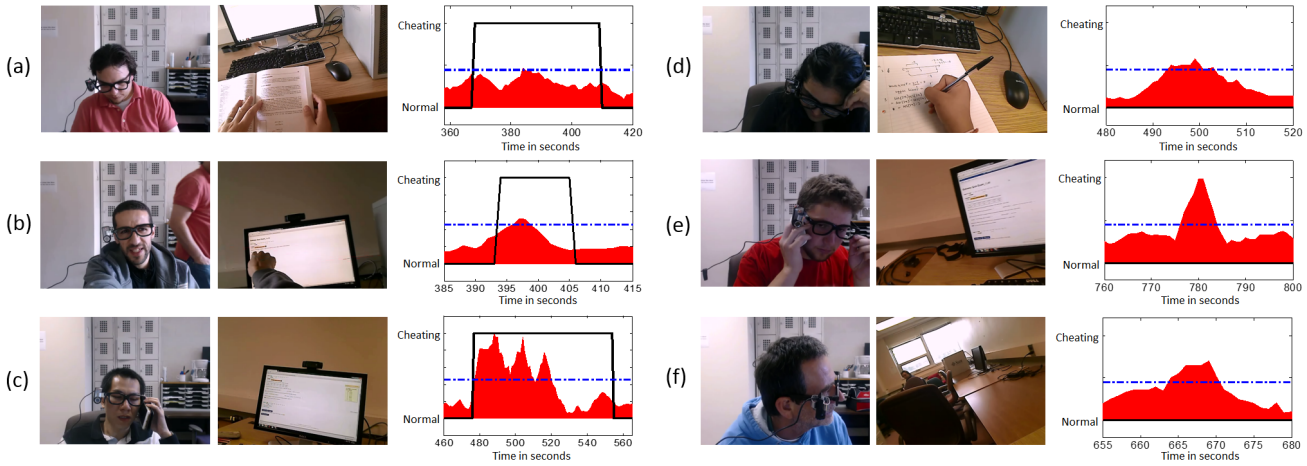

Fig. 16: Failure examples of the OEP system, showing the frames from both of the webcam and wearcam along with the estimated cheat behavior probability for a specific duration in the test taking illustrated by the x-axis. (a, b, c) represent cases where the OEP struggles to recognize the cheat activity 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.

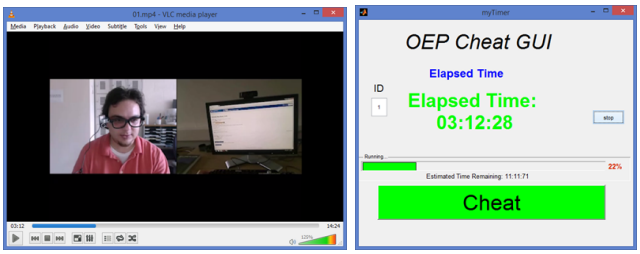

Fig. 17: The GUI for human proctoring used by three proctors.

		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.

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.

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 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 $\sim$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.

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.

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.

[10] H. Hung and D. Gatica-Perez. Estimating cohesion in small groups using audio-visual nonverbal behavior. *IEEE Trans. Multimedia*, 12(6):563–575, 2010.

[11] 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.

[12] A. Jourabloo and X. Liu. Pose-invariant 3d face alignment. In *Proc. Int. Conf. Computer Vision (ICCV)*, pages 3694–3702, 2015.

[13] I. Jung and H. Yeom. Enhanced security for online exams using group cryptography. *Education, IEEE Trans. on*, 52(3):340–349, 2009.

[14] 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.

[15] D. L. King and C. J. Case. E-cheating: Incidence and trends among college students. *Issues in Information Systems*, 15(1), 2014.

[16] I. Lefter, L. J. Rothkrantz, and G. J. Burghouts. A comparative study on automatic audio–visual fusion for aggression detection using meta-information. *Pattern Recognition*, 34(15):1953–1963, 2013.

[17] 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.

[18] X. Liu. Video-based face model fitting using adaptive active appearance model. *Image and Vision Computing*, 28(7):1162–1172, July 2010.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] J. Roth, X. Liu, and D. Metaxas. On continuous user authentication via typing behavior. *IEEE Trans. Image Process.*, 10:4611–4624, Oct. 2014.

[26] 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.

[27] 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.

[28] M. Savvides, B. V. Kumar, and P. Khosla. Face verification using correlation filters. *3rd IEEE Automatic Identification Advanced Technologies*, pages 56–61, 2002.

[29] J. Shi and C. Tomasi. Good features to track. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, 1994.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

View publication stats