Blackboard Usage Detection and Segmentation for Presenter Tracking Systems

Charles Fitzhenry
Honours, Department of Computer Science
University of Cape Town
Rondebosch, 7701, South Africa
ftzcha002@myuct.ac.za

ABSTRACT

Lecture recordings have recently become an effective, and popular, learning tool. This research forms part of a larger project for developing a lecture tracking system for capturing lectures.

We present and evaluate a system that uses computer vision to detect when blackboards are used and then segment the tightest window enclosing them. This is done by detecting each individual board and then building the large cropping window from this information. The study contributes to a project that is run by the Centre for Innovation in Learning and Teaching (CILT), that implements lecture recordings at the University of Cape Town (UCT).

Our results showed that we were able to successfully detect board usage in less than 90% of the input video's length. This means that if an input video is 10 minutes long, our program is able to complete processing in less than 9 minutes. This satisfies the design constraints set out by CILT.

CCS Concepts

• **Computing methodologies → Computer vision problems**

• **Computing methodologies → Video segmentation**

• **Computing methodologies → Tracking**

Keywords

Presenter Tracking; 4K Video; Light Normalization; Blackboard Tracking; Blackboard Segmentation

1. INTRODUCTION

Recently the video recording of lectures has become popular and has been implemented as an accepted practise at many universities and institutions worldwide [4, 5]. It has been implemented widely across Massive Open Online Courses (MOOCs), making distance learning very accessible and engaging [21]. There are many different terms used to refer to the concept of lecture capturing, but the general definition interprets it as the use of any technology to record what a lecturer does and says during a class, and then making this digitally available to students [15]. Specialised software and hardware are used to capture the lecture as well as any other presentation medium such as presentation slides used during the lecture. Since the lecturer does not have to wear specific equipment to be detected by the camera there is a need to utilise methods from the field of computer vision, most notably tracking, based on the video source only [21].

Vassar *et al.* [15] and Al-Nashash *et al.* [1] discuss the importance of lecture recording and how it eliminates the need for students to make detailed notes in class. In doing so, students can engage more actively in the class and grasp the material that is being explained. Furthermore, it significantly assists students with language barriers and learning disabilities as they can review the lecture again at their own pace and facilitates collaboration between students. Bernal *et al.* [3] investigate the role of audio-visual aids in the context of psychology lecture. Their results showed that 38.88% of students prefer taking notes from the boards, while 42.22% prefer taking notes from the PowerPoint slides. They also reiterate the benefits of lecture recordings and note that the benefits vary depending on the content and the objective of the class. Since the content written on the board needs to be legible in the recorded videos, it becomes essential to maintain focus on the lecturer while maximising the view of the content written on the boards.

The Centre for Innovation in Learning and Teaching (CILT) is responsible for implementing lecture recording at the University of Cape Town (UCT). The recorded lectures are made available to students on a Sakai-based classroom management system known as "Vula". To date, CILT has tested three different approaches to recording lectures. These are detailed below:

Static Camera

This approach utilises a camera with a static field of view that captured a set section of the lecture area. Often this approach does not capture the entire area, especially in very wide venue configurations and hence information is lost. Because it captures the entire region, it also makes reading from the boards more difficult. This is the cheapest option costing R10 000 for a camera.

Pan-Tilt-Zoom (PTZ) Camera

This approach uses a PTZ camera along with a raspberry pi which has a small overview camera. The raspberry pi runs the LectureSight [21] software, which tracks the lecturer in real time and then drives the PTZ camera to follow the lecturer. This approach works better than the static camera, however, because tracking is done in real-time, and the lecturer may move in an unpredictable manner, the system may lose track of the lecturer or lag behind when they start walking. This is the most expensive approach: the total setup including the raspberry pi amounts to the cost of R48 000.

4K High-Resolution Camera

This approach is the latest one being tested by CILT and it uses a very high resolution (3840 x 2160 pixel dimensions) 4K camera which captures a very wide angle of the lecture area. This video is then stored and needs to be processed after the lecture, to reduce the size of the video. This approach is cheaper than the PTZ approach and costs R17 442 for a camera.

Images of these cameras can be seen in Figure 1 below.

Figure 1 - From Left: Static Camera, PTZ Camera, 4K Camera

The problem with CILTs latest approach is that the videos files produced by the 4K camera are very large (1.97GB per 50min lecture using the MP4 CODEC) and cannot be streamed via Vula.

This project forms part of a larger project that aims to develop a computer vision system to virtually track the lecturer in the 4K video. In particular, this project aims to investigate the detection of boards and when they are used as well as segmenting/cropping only the region in which the boards are. This information will then be able to be used to make better panning adjustments. The panning system is the subject of another paper.

CILT has stated that their expectation of this project is that the blackboards are legible and to avoid a video post-processing backlog from forming, videos must be processed within 8 hours of recording and each video may not take more than 300% of its length for processing. With these constraints we need to know the following to determine if the project is a success:

1. Can we detect blackboard usage in a lecture recording video stream using computer vision algorithms?

2. Using only modest computer hardware, can the required processing be completed in a time that is less than 90% of the video length?

The results of the evaluation show that it is possible to detect the board usage and that the entire process can be completed in 88% of the input video's length.

This paper is laid out as follows. Section 2 examines related work, Section 3 discusses the design and implementation, Section 4 discusses the results and methods and Section 5 concludes and discusses future work.

2. RELATED WORK

2.1 Board detection

Onishi et al. [9] proposed a method for segmenting regions of text from a blackboard by using recorded lecture videos. They identify that it is important to capture a shot of the blackboard that is legible to students and they also note how conventional approaches show either the lecturer or the whole blackboard only, making it difficult to see what is written on the board. Writing on the blackboard is detected by using edge detection (Sobel) functions. The algorithm incrementally pans the section as the lecturer adds new text to the board. The experimental results show that the technique achieves accurate segmentation results in 91.5% of the time. Liu et al. [7] suggest an improvement that does not assume a static board, but instead can accommodate moveable boards under varying lighting conditions. They also consider that the board colour can change due to the accumulation of chalk dust. Their method can also be applied to whiteboards. They model the colour distribution of the board and this solves the challenges faced by techniques that assume a uniform background colour of the board. Wallick et al. [17, 18] independently proposed similar techniques to [7] and note that the problem of segmenting blackboard images does not require the creation of new fundamental computer vision techniques, but instead using already existing techniques in a way that would achieve desirable results. Onishi et al use an edge detection algorithm to detect the boards in their implementation. This approach of using an edge detector was used in our approach too. However, we implemented a canny edge detector and the reasoning will be explained in a later section.

2.2 Illumination Correction

Histogram Equalization (HE) is a technique used to adjust the intensities of images to enhance contrast. The HE technique has been referenced by Hummel [6] as early as 1974 and although the original HE algorithm was only able to correct the lighting of grayscale images, the algorithm has been extended and improved extensively since then. In 1992 Trahanias and Venetsanopoulos [14] successfully extended the HE algorithm to colour images, in a technique called 3-D Histogram Equalization. This proposed method could not be measured quantitatively because the evaluation criteria for the performance of such a method depends on the visual appearance of the final image produced. However, the method successfully enhanced the images without producing any colour artefacts as opposed to other similar techniques. Although this method has been shown to produce good results, one significant drawback is the computational complexity, that is $O(n^3)$ where n is the discreet number of grey levels (or bins) in the histogram [14]. Naik and Murthy [8] identified a problem with this approach that shifts the three primary colour components Red, Green, and Blue (RGB) unequally, resulting in a change of the pixel's hue. They proposed an improvement that extended to colour images and preserved the hue during correction, and eliminated the gamut problem, where pixels go beyond the displayable bounds after the process. We were unable to implement normal HE because of the lighting being unevenly distributed. When normal HE is applied, the dark areas in an image are correctly enhanced but the bright areas become overexposed. This problem is solved by an Adaptive Histogram Equalization (AHE) technique that works similar to HE, but instead of applying the HE algorithm to the entire image (global), the images are divided into smaller regions to form a grid over the image. The HE is then applied to each region individually and is repeated for all regions in the image. One problem is that should two regions adjacent to each other have very different contrasts, this algorithm will re-map each region very differently, which may cause a seam to appear between the two regions. This is resolved by limiting the contrast allowed by the HE algorithm and this combined technique is known as the Contrast Limited Adaptive Histogram Equalization (CLAHE). The seams are also interpolated to prevent any indication of their existence [11].

Figure 2 illustrates the difference between HE and CLAHE.

Figure 2 - From Left: Original Image, HE, CLAHE

The latest contributions and research to this technique are from Wong et al. [20] and Wang et al. [19]. Wong et al. indicates the shortcomings in all the current approaches, that is, they often produce undesirable results caused by over-enhancement and when limited to only intensity-enhancements, the colour and saturation aspects of the image do not get enhanced. To solve this,

a pipelined approach that incorporates colour stretching, HE, magnitude compression and saturation maximization, is proposed. The experiments show that this algorithm solves all the previous challenges associated with HE without affecting performance [20]. Wang et al. take a slightly different approach to achieve the same results. In previous HE approaches, histograms were constructed and a mapping was determined, upon which the pixels of the source image were then adjusted. This new approach instead aims to adjust the image pixel values directly and the resultant histogram can be redistributed [19].

3. DESIGN AND IMPLEMENTATION

3.1 System Constraints

CILT has provided the following constraints to which the system should adhere:

- System should run on Linux (Ubuntu)
- Not exceed the capacity of their 24 CPU blade servers with 96GB RAM available and should be able to run on their Virtual Machines (VMs) which typically have 4-8 CPUs and 4-16GB RAM.
- Preferably not use the GPU for performance enhancements
- The entire system should complete the video processing in less than three times the videos runtime (i.e A 45-minute-long video should be processed in less than 135 minutes).
- Use multi-threading to optimise process

3.2 Requirements

3.2.1 Functional Requirements

The system needs to take in a 4K video and perform the following:

- Detect areas of motion
- Detect boards
- Detect board usage
- Segment boards and output to a separate video stream

3.2.2 Non-Functional Requirements

The following requirements do not directly affect the functionality of the system, however, they are essential in ensuring that the system performs optimally and need to account for the constraints set out by CILT.

- **Compatibility** - System needs to be compatible with Linux.
- **Efficiency** – Need to process video in less than three times its length.
- **Aesthetics** – The output should be visually appealing and benefit the viewer.

Amongst the goals of this project are to ensure that the writing on the boards is sufficiently legible and that the panning process can optimise the view to include the lecturer and as much of the board being used as possible. Because the segmented boards will form a separate video stream, and its primary purpose will be to allow students to see written content on boards at all times, the frame rate of the segmented stream will be reduced to 1 frame per second to reduce the size of the video.

3.3 Design

The system was implemented using C++ and the OpenCV [10] library. The OpenCV library was chosen because it is an open source library which provides a comprehensive set of optimized computer visions and image processing functions. This allows our project to focus on achieving its functionality goals, instead of to re-implementing functions that already exist. Because it is open source it does not add any additional costs and the code of our project can be made available for future development.

The system, titled Natural Presenter Tracking in 4K Video (TRACK4K) takes a 4K video as input and processes it in the following stages:

- **Board detection and segmentation –** The boards are identified, segmented and written to a separate video.
- **Lecturer tracking** – The lecturer is identified and tracked in the video.
- **Virtual Cinematographer** – The lecturer is cropped into a smaller frame and this is saved to an output video.

Board detection and segmentation is the focus of this paper. Figure 3 shows a high-level flow of the system and how the components are integrated.

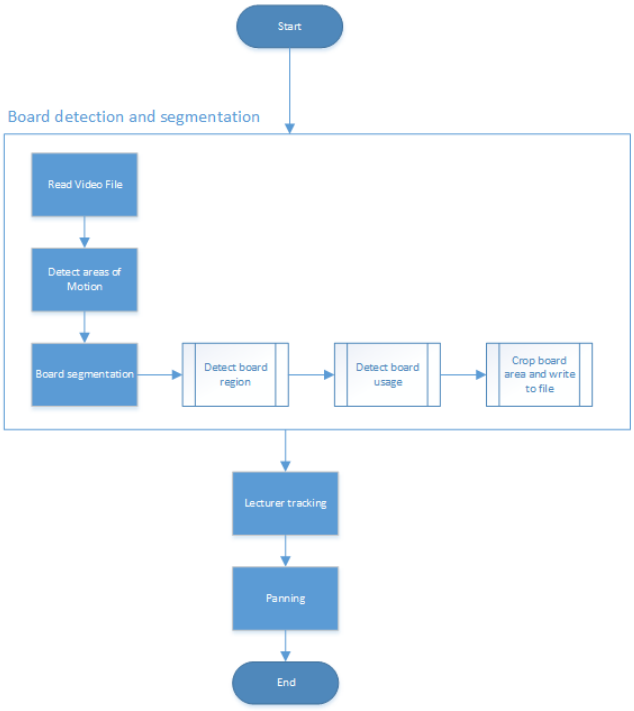

Figure 3 - System Overview Diagram

3.3.1 Illumination Correction

The board detection method detects features in the frames based on their characteristics such as colour, brightness, size and other heuristics. Features are locations in an image such as corners blobs and T-junctions [2]. For feature detection to work reliably, the input frames from the video must have sufficient brightness and/or contrast. The videos are recorded in different venues with different light, window and board configurations and the lecturer may also alter the light brightness or close the blinds or the windows in the venue. All these factors manifest in the videos and because their lighting conditions differ, may cause the detection algorithms to produce inconsistent results. To combat this problem, an illumination correction algorithm is applied to the

frame that is to be used by the board detection method. Initially in early implementations of the project, the illumination correction was applied to every frame in the video file, however, since the board detection does not evaluate each consecutive frame, the illumination correction only needs to be done on the required frame. This resulted in a dramatically faster processing time and memory efficiency.

3.3.2 Basic Motion Detection/Segmentation

This functionality detects where all motion occurs over a set number of frames and then bounds these regions with rectangles. This is useful for the tracking stage of the project as it reduces the search space when looking for the lecturer. It uses the idea of subtracting frames to find the difference between them and then combines all the resultant differences into one final image representing all motion over those frames. An illustration of this can be seen in Figure 7.

3.3.3 Blackboard Segmentation and Usage Detection

The board segmentation detects the boards in the venue and finds the tightest bounding box that includes all the boards. This crop region is then saved with a low frame rate of one frame per second. The primary purpose is to focus on the writing on the boards as required by CILT.

To be able to find the board regions we need to look for shapes in the frame and this is achieved by using an edge detector to detect the boundaries of the boards and the board columns which can then be used to segment the boards. The method used to detect edges is a Canny Edge detector. Canny was chosen over a Sobel edge detector since it is considered to be significantly more accurate and slightly more efficient in terms of execution time [16].

To be able to detect when the board is used, we needed some way of determining how much of the board is used and to detect when this changes. To do this we use a feature detection algorithm that returns the number of key points such as corners and edges which would typically be associated with writing. The SURF function was used instead of the Scale Invariant Feature Transform (SIFT) as it has been shown to outperform SIFT and produces more robust results. These functions can be used to identify features between two images and attempt to match features [2]. This matching feature was however not used in this implementation, as we only needed the number of features, thus reducing computation time.

3.4 Solution Overview

The system opens a video file and then proceeds to read a pre-determined number of frames into memory. Each read iteration performs the following operations on the video segment that was read into memory. The process is to then identify all areas in which motion occur over this segment of the video. This information would be useful for the actual lecture tracking stages (not in the scope of this project) as it would allow the search space to be reduced to the areas that only contain motion. Secondly, the boards need to be detected and when they are used. This is done by looking through the video segment and detecting objects that meet a predefined set of features that would best describe a board. These areas are then boxed into rectangles and evaluated for features and when the feature count begins to increase beyond a certain threshold (to account for false positive cases) that rectangle is flagged as being used and it is the evaluated in which half of the screen it is, left or right. This usage information is used for the panning section to ensure optimal pan (out of the scope of this paper). During the board detection stage, it is necessary to correct the lighting, as some videos have varying lighting conditions which can affect the detection stage. The group of processes repeats until no more frames are left in the video stream. Once all of these processes are complete, the information is stored in one central class that will be shared between the other sections, namely lecturer-tracking and panning, which is both out of the scope of this paper.

3.5 Implementation

This section discusses the implementation of the system and how the functionality has been achieved. Figure 4 below shows a class diagram of the board detection.

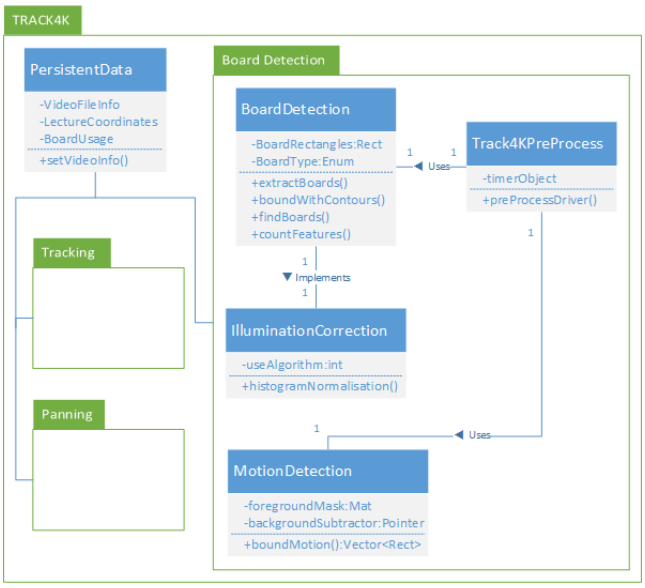

Figure 4 - Board Detection Class Diagram

3.5.1 Illumination Correction

Lecture venues have an overhead light directly above the board region which concentrates the light onto the boards and becomes dimmer towards the sides and the bottom of the video. This caused problems during initial implementation and we use the CLAHE [11] method from OpenCV with a clip limit parameter set to 1, to correct the lighting. It was found that a value higher than one resulted in a grainy image and produced a poorer result when detecting the boards.

3.5.2 Basic Motion Detection/Segmentation

This method is implemented by iterating over the frames in each video segment. The absolute difference is taken between two frames and then passed through a threshold function which converts it into a binary image. The absolute difference function takes in two images as input and then outputs a new image with each pixel value being the pixel value of image one minus the pixel value of image two. The absolute value is taken on this difference to ensure that only positive values are returned [22]. A threshold function works on a grayscale image by evaluating the intensity of each pixel and if it exceeds a specified threshold value it is set to be 255 (full white) else it is set to be 0 (full black) and hence a binary image is formed [12]. An overview of this process can be seen visually in Figure 5 and in pseudo code in Code Block 1. This resultant binary image is then accumulated by using a bitwise-or function to finally achieve the overall areas of motion as seen in Figure 7. Figure 6 illustrates how the lecturer has

moved in the venue which has led to the resultant image in Figure 7.

```
for every frame:
        Convert image to grayscale
        Subtract frame i from frame i+1
        Threshold result
        Add difference to final resultant image
```

Code Block 1 - Code to generate merged motion

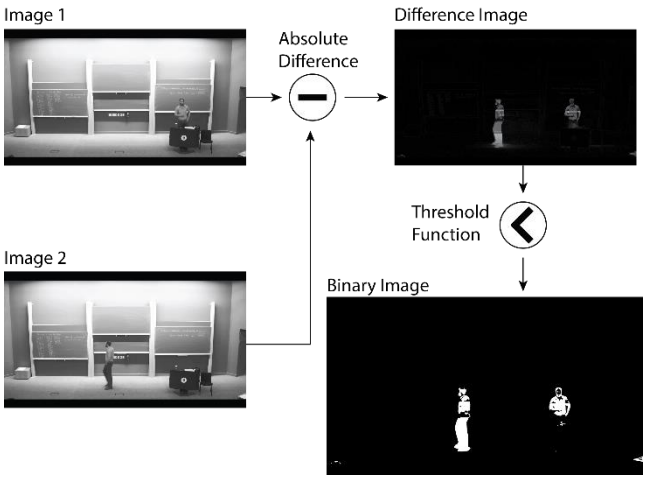

Figure 5 - Generation of binary image

Once the final binary image with all the motion has been generated, it is passed to the OpenCV findContours() method which finds contours in a binary image and outlines shapes with a close approximation [13]. Each contour is returned as a vector of points which can then be passed to the boundingRect() method that calculates the tightest rectangle to enclose the set of points. This stage yields the rectangles as seen in Figure 6 and these rectangles are then stored in a 2D vector to be passed on to the tracking stage.

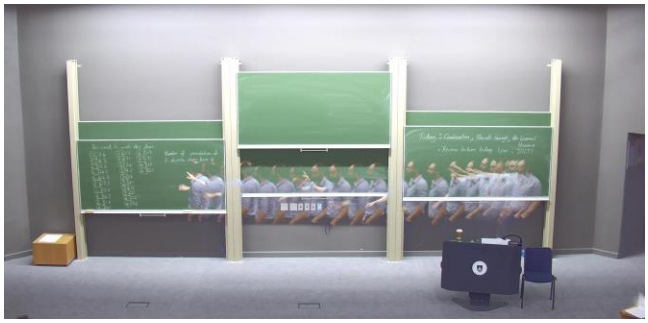

Figure 6 - Superimposed Frames

Figure 7 - Binary Image Showing all Motion Merged and Bounded with Rectangles

3.5.3 Blackboard Segmentation and Usage Detection

The algorithm works by iterating over every 29th frame and then passes this frame to a function that detects rectangles in the image. This rectangle finding algorithm works by the following stages as seen in Code Block 2.

```
for every 29th frame:
        Convert image to grayscale
        Find contours
        Bound contours with rectangles
        Classify rectangles as board types

        for every board:
                Count features
```

Code Block 2 - Board Classification

Once all rectangles in the frame have been classified, these rectangles are then passed into a feature detection function. We needed a way of detecting when the boards are being used and by using a feature detection algorithm, we can count the features and whenever we detect an increase we can consider the board being used. The Speeded Up Robust Features (SURF) function was used to count the number of key points in each rectangle. In particular, the board rectangles are of interest as this is used to detect when they are used. The boards are identified on every 29th frame and the number of features is also counted at the same time and stored for the usage detection. We find that using 1 second (29 frames) intervals to evaluate usage produced better results as it allowed the lecturer enough time to add enough writing to be able to detect that the board was in use. During the usage detection, we iterate over all the stored frames containing the board locations. On each iteration, we evaluate the number of features by tallying up the total count that occurs in the left half of the screen and then the right. So we only consider left or right columns because when the lecturer is using the centre board, the panning module will frame him in the centre and does not require an offset. Then once tallied per side, we evaluate each new iteration and if the new count minus the count from the previous iteration is greater than a threshold value of 150, we assume the board was used. 150 was chosen, because we found that a value between 70 and 150 detected the board even when the lecturer passes in front of it without writing on the board. A value greater than 150 was too sensitive and the lecturer needs to write a lot before usage is detected. Depending on which side of the board is used, the left or right is flagged as being used for that frame. This count is then re-evaluated on each new frame.

Figure 8 shows a rectangle with writing on the blackboard and shows how the SURF function detects features. Figure 9 shows how the SURF function does not detect features when the board has no writing.

Figure 8 - Blackboard with features (Writing on board)

Figure 9 - Blackboard with very few features (No writing on board)

4. RESULTS

In this section, we discuss the methods that were used to test and evaluate the system.

4.1 Methods

The system has been tested using the following computer specifications:

CPU: Intel Core i7-6700HQ CPU @ 2.60 – 3.5GHz

RAM: 16GB DDR4 2133MHz

Hard Drive: Samsung 850 EVO SSD

OS: Windows 10

A set of test cases was created to test both the ability to detect the overall bounding box to crop the boards as well as the usage detection algorithm. For the test cases, all videos used had a frame rate of 29FPS.

4.1.1 Board Segmentation Test Cases

To test the detection of the boards in the venue, it is important to consider the various use cases that can occur. Because the detection algorithm looks for edges in the image, it is necessary to test cases where the boards are obstructed partially or fully and because lighting affects this process, different light conditions are also tested. These test cases were selected based on watching several lecture videos and observing the behaviour and usage of the boards. Lecturers use different light configurations, however, none of the videos watched ever had a lecturer using the blackboards while the lights were entirely off. The only time the lecturers switched the lights off were when they used the data projector. The test cases also consider the usage of the projector and how this affects the overall board detection. Table 1 below summarises these test cases.

Table 1 - Board detection test cases

Test Number	Board Configuration	Lights
1	All boards visible	ON
2	All boards aligned behind one another so only one is visible per column	ON
3	Projector screen down	ON
4	Projector screen down and projector on	OFF

To perform these tests, a custom video was recorded since we did not have enough lecture videos and none of them had all the cases that were required. The recorded videos were recorded in lecture venue CS2A in the Computer Science building at UCT, using the built-in 4K camera. To perform the tests on this video, segments were cut out and limited to 580 frames. This value was chosen because the detection algorithm detects every 29th frame. This allows us to ensure that a large enough sample of frames was evaluated to increase confidence in the board detection.

4.1.2 Board Usage Detection

To test the usage of the boards in a venue we assume that the lighting is generally good. This has been noted through watching many lecture recordings and discovered that lecturers always have either the room lights on full brightness and/or the board lights on. Because the board detection works by detecting the number of features on each board and storing this usage as being either on the left side of the venue or the right, it is important that the tests focus on these board columns more than the central board column. The test cases look at how the algorithm performs in the cases in Table 2. The table also shows the expected outcome for each test.

Table 2 - Board usage detection test cases

Test Number	Test Case Description	Expected Outcome
1	Lecturer walks passes board without writing on it	Not used
2	Lecturer gestures in front of the board (Hand movements, pointing, but not writing)	Not used
3	Lecturer writes on board	Used
4	Lecturer moves board with writing on	Not Used
5	Lecturer moves empty board	Not used
6	Lecturer erases board	Used

The tests for the board usage detection was performed on a real lecture recording. This was done to get the most natural use cases and the test cases only needed to test common board usage behaviour. The video was cut into sections that covered the test cases and the number of frames and frames examined the results are displayed.

4.1.3 Runtime

The runtime of the program is evaluated by starting a timer before the method call for the module and then stopped after it has completed. The time of this is then printed out.

4.2 Data

The data for both sections 4.2.1 and 4.2.2 were obtained by manually stepping through the program in debug mode and counting the results. The data for the runtime was collected by printing the time out to the console in the program.

4.2.1 Board Segmentation

The data represented is divided by board column in the lecture venue and a result of 0/1 means that there were 0 boards detected when in fact there was one board that should have been detected. Cases 3 and 4 involved testing the lecturer screen in the left column only. The data can be seen in Table 3.

Table 3 - Board detection results

Test Number	Left Column	Centre Column	Right Column
1	2/2	1/2	1/2
2	1/1	0/1	1/1
3	Screen detected	0/1	1/1
4	Screen not detected	0/1	0/1

4.2.2 Board Usage detection

The data for the board usage detection displays the number of frames that were supplied as input. These vary in length because the different actions by the lecturer take different times, for example, moving a board is a fast operation and hence has fewer available frames for testing. Because every 29th frame is evaluated (hence forth known as an instance) by the algorithm, at least two instances in a frame collection need to be available, and in this data, there were at least 8 instances for each case.

Table 4 - Board usage detection results

Test Number	Total Input Frames	Instances Evaluated	Instances of Board Usage
1	232	8	0/8
2	743	25	0/25
3	1082	37	6/37
4	248	8	0/8
5	282	9	0/9
6	379	13	0/13

4.2.3 Runtime

The data for the runtime was achieved by processing a 50-minute-long lecture recording. Each operation was reported individually and then totalled for the entire operation. We noticed that the runtime was independent of board usage or lecturing style because the algorithms have to perform the evaluations regardless.

Table 5 - Runtime results

Operation	Time taken
File read	31m58s
Motion Detection	2m13s
Board Segmentation and usage Detection	10m37s
TOTAL	44m48s

4.3 Discussion

Looking at the data in Table 3, the algorithm correctly identified the boards 80%, 20% and 60% of the time for the left, centre and right columns respectively. Because the segmentation algorithm uses the left and right column to produce the cropping region, it is important that at least one board is detected in this case. The centre board performed very poorly and this is likely as a result of the board having been very dusty and white as opposed to the others. Because the detection depends on the darkness of the boards, this would almost certainly have been the reason for such a poor performance. Test case 4 failed entirely and this could be expected, since there was no lighting in this case and was an extreme to determine the limitations of the algorithm and in most cases we can assume that this would not happen and at least not for the entire lecture. The algorithm only needs a few frames with good lighting to detect the crop region and because the video is post-processed, we are able to look ahead through the video until the crop is found.

The board usage detection algorithm produced very good results with the exception of test case 6. We had expected that erasing the board would be detected as board usage; however out of the 13 evaluations, none returned a significant change to trigger a board usage event. Because the algorithm works by counting the features, shifting the boards would not trigger a change. The data also shows a low detection while the lecturer is writing, however, this is because the lecturer does not write continuously and when writing a new word, the pause in between will not be considered. This is likely why only 6 of the 37 frames evaluated contained usage information. This is not significant since it is only required to detect at least one usage event because these will be used in the panning module by attaching it to a single pan operation, which stretches over a few seconds.

The runtime of the code is shorter than the length of the input video and completes in just 88% of the input video's time. This very closely reaches the research question of whether it would be possible to process the videos in less than 90% of the input video's length. These results can be seen in Table 5 and a visual distribution of the time on the different sections can be seen in Figure 10.

We sent the output video of the cropped boards to Stephen Marquard (Coordinator of Learning Technologies Portfolio) at CILT and he had the following feedback regarding the video:

- Video can be cropped more than it is at present
- It would be interesting to see whether perspective correction is valuable, i.e. adjust the picture in such a way that all the blackboards are rectangles.
- Not sure if there's value in segmenting into individual boards, given that they move up and down, and also that there's value in seeing everything at the same time
- It might be interesting to see if one could identify segmentation points in the video (i.e. like bookmarks), at the points where the lecturer starts writing on a different blackboard. So you could quickly get the whole content of the lecture board-by-board by jumping

to 6 points or more if blackboards are reused (cleaned and written over).

- Experimentally one could test the impact of different fps rates on the overall file size and also check user acceptability to see what a good trade-off is (e.g. 1fps vs 5ps vs 10fps).
- Overall this is a good technique to provide users with more choice about how to consume video of blackboard-based lectures depending on their context (e.g. users with a large desktop monitor may choose this option, but if you're watching on a small screen device like a smartphone then you may choose the tracked option).

It is possible to crop the video more. This can be done by looking at the boards only and not their containing columns to determine crop region. Since all blackboards are already segmented individually, it would be possible to pass this through a keystone correction algorithm to correct the perspective. Adding the segmentation points in the video would be possible to implement considering that the boards are already evaluated for features and this could be used to skip through the video.

5. CONCLUSIONS

We needed to develop a system that would successfully be able to segment the blackboards while detecting the board usage. To solve this problem, computer vision algorithms were used to detect the board usage in an input lecture video and the following conclusions were made.

It is possible to detect blackboard usage in a lecture recording using computer vision algorithms. Although our system struggled to detect the boards in low-light conditions as well as when the boards are very dusty, the algorithms are robust in the sense that they only need the detection to occur at some point through the video, to correctly segment the board region.

We also found that it is possible to process the video in time less than 90% of the original video's length. This satisfies the requirement from CILT that the processing time should not exceed 300% of the original video length (or 100% for this module).

5.1 Future Work

In future work, it would be important to address the robustness of the detection algorithms, especially in the context of varying light conditions as well as board conditions. This project only considered blackboards and projector screens and it would be beneficial to develop a system that will detect the type of board first and then proceed based on that. This will ensure that if lecture venues are fitted with whiteboards instead, the system will still detect the boards correctly. It would be useful to segment the completed boards as individual components by using the past and future information to segment the board when the lecturer moves onto another board and is not obstructing the board to be segmented. This could then be saved as a collage of boards as currently only a tight crop is used to remove excess space around the board area.

6. REFERENCES

[1] Al-Nashash, H. and Gunn, C. Lecture Capture in Engineering Classes: Bridging Gaps and Enhancing Learning. Educational Technology & Society, 16, 1 (January 2013), 69-78.

[2] Bay, H., Tuytelaars, T. and Van Gool, L. Surf: Speeded up robust features. In Anonymous *European conference on computer vision*. Springer, 2006, 404-417.

[3] Bennal, A., Itagi, V. and Taklikar, R. Role of audio-visual aids in physiology lecture. Natl J Physiol Pharm Pharmacol, 4, 2 (2014), 109-111.

[4] Demetriadis, S. and Pombortsis, A. E-lectures for flexible learning: A study on their learning efficiency. Educational Technology & Society, 10, 2 (April 2007), 147-157.

[5] González-Agulla, E., Alba-Castro, J. L., Canto, H. and Goyanes, V. Galitracker: Real-time lecturer-tracking for lecture capturing. In Anonymous *Multimedia (ISM), 2013 IEEE International Symposium.* (Anaheim, CA, December 9-11,2013). IEEE, 2013, 462-467.

[6] Hummel, R. A. Histogram modification techniques. Computer Graphics and Image Processing, 4, 3 (September 1975 1975), 209-224. DOI=http://dx.doi.org.ezproxy.uct.ac.za/10.1016/0146-664X(75)90009-X.

[7] Liu, T. and Kender, J. R. Lecture videos for e-learning: Current research and challenges. In Anonymous *Multimedia Software Engineering, 2004. Proceedings. IEEE Sixth International Symposium on*. IEEE, 2004, 574-578.

[8] Naik, S. K. and Murthy, C. Hue-preserving color image enhancement without gamut problem. Image Processing, IEEE Transactions on, 12, 12 (December 2003), 1591-1598. DOI=10.1109/TIP.2003.819231.

[9] Onishi, M., Izumi, M. and Fukunaga, K. Blackboard segmentation using video image of lecture and its applications. In Anonymous *Pattern Recognition, 2000. Proceedings. 15th International Conference on.* (Barcelona, September 30, 2000). IEEE, 2000, 615-618.

[10] OpenCV. (2016). DOI=http://opencv.org/.

[11] Reza, A. M. Realization of the contrast limited adaptive histogram equalization (CLAHE) for real-time image enhancement. Journal of VLSI signal processing systems for signal, image and video technology, 38, 1 (2004), 35-44.

[12] Sezgin, M. Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic imaging, 13, 1 (2004), 146-168.

[13] Suzuki, S. Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing, 30, 1 (1985), 32-46.

[14] Trahanias, P. and Venetsanopoulos, A. Color image enhancement through 3-D histogram equalization. In Anonymous *Pattern Recognition, 1992. Vol. III. Conference C: Image, Speech and Signal Analysis, Proceedings., 11th IAPR International Conference on.* (The Hague, August 30-September 3, 1992). IEEE, 1992, 545-548.

[15] Vassar, P., Havice, P. A., Havice, W. L. and Brookover, R. The Impact of Lecture Capture Presentations in a Distributed Learning Environment in Parks, Recreation, and Tourism Management. Schole, 30, 1 (2015).

[16] Vijayarani, S. and Vinupriya, M. Performance analysis of canny and sobel edge detection algorithms in Image Mining. Int.J.Innovative Res.Comp.Commun.Eng, 1, 8 (2013).

[17] Wallick, M. N., Gleicher, M. L. and Heck, R. M. Obtaining a Mid-level Representation of Handwriting without Semantic Understanding. (2003).

[18] Wallick, M. N., Heck, R. M. and Gleicher, M. L. Marker and chalkboard regions. In Anonymous *Proceedings of Mirage*. 2005, 223-228.

[19] Wang, W., Chen, C. and Ng, M. K. An image pixel based variational model for histogram equalization. Journal of Visual Communication and Image Representation, 34(January 2016), 118-134. DOI=http://dx.doi.org/10.1016/j.jvcir.2015.10.019.

[20] Wong, C. Y., Jiang, G., Rahman, M. A., Liu, S., Lin, S. C., Kwok, N., Shi, H., Yu, Y. and Wu, T. Histogram Equalization and Optimal Profile Compression based Approach for Colour Image Enhancement. Journal of Visual Communication and Image Representation, (April 2016). DOI=http://www.sciencedirect.com/science/article/pii/S1047320316300529.

[21] Wulff, B. and Fecke, A. Lecturesight-an open source system for automatic camera control in lecture recordings. In Anonymous *Multimedia (ISM), 2012 IEEE International Symposium on*. (Irvine, CA, December 10-11, 2012). IEEE, 2012, 461-466.

[22] Zivkovic, Z. Improved adaptive Gaussian mixture model for background subtraction. In Anonymous *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*. IEEE, 2004, 28-31.