A Virtual Cinematographer for Presenter Tracking in 4K Lecture Videos

Technical Report CS17-01-00

Department of Computer Science

University of Cape Town

2017

Charles Fitzhenry; Maximilian Hahn; Tanweer Khatieb; Patrick Marais; Stephen Marquard

Corresponding author: patrick@cs.uct.ac.za

ABSTRACT

Lecture recording has become an important part of the provision of accessible tertiary education and having good autonomous recording and processing systems is necessary to make it feasible. In this work, we develop and evaluate a video processing framework that uses 4K video to track the lecturer and frame him/her in a way that simulates a human camera operator. We also investigate general issues pertaining to blackboard usage and its influence on cinematography decisions. We found that post-processing produced better tracking and framing results when compared to some real-time approaches. Furthermore, the entire pipeline can run on a commodity PC and will complete within the suggested time of 300% of the input video length. In fact, our testing showed that 60% of the total processing time can be ascribed to I/O operations. With the removal of redundant reads and writes, this proportion can be reduced. Finally, some algorithms can be remapped to parallel versions which will exploit multicore CPUs or GPUs if these are available.

CCS CONCEPTS

• **Computing Methodologies** → **Computer vision problems**

• **Computing Methodologies** → **Video segmentation**

• **Computing Methodologies** → **Tracking**

KEYWORDS

Background Segmentation; Lecture Recording; Movement Detection; Object Tracking; Presenter Tracking; OpenCV Library; Virtual Cinematography; Panning.

1 INTRODUCTION

The recording of lectures has become popular in the last decade and has been implemented at universities and institutions worldwide [1, 2]. In this work, we focus on the technical aspects of preparing a captured video stream for online consumption.

Course content written on the boards is important [2–5], and it is necessary to capture this when filming the lecturer and to ensure that it is clearly legible. It is also important that the video is filmed in a way that is natural, with the camera tracking the lecturer and keeping the board in view for context, when appropriate. Another important requirement is that the size of the video or the bitrate required to stream the video should be such that a student with a modest internet connection can benefit from the recording. This is particularly important in the developing world where network infrastructure is often poor. The recent adoption of 4K video cameras to record lectures has exacerbated this problem.

Furthermore, the existing solutions that are commonly used to record and prepare videos are often proprietary and have a high cost, which may further hamper adoption in developing countries.

A 4K camera has a resolution of 3840x2160. This provides a wide viewing angle allowing the camera to capture a large portion of the scene. Although this produces high-quality videos, the file size is so large (bit rate of 5.652Mbps) that it is inaccessible to many students and institutions that cannot afford the necessary data bandwidth to stream these videos in 4K. An alternative approach is to crop a smaller window of "attention" from the larger 4K frames. By using computer vision algorithms to detect and track the lecturer, we can move this cropping window to follow the lecturer and frame relevant context.

Our work aims to address these issues by *post-processing* of 4K video streams to produce an output video that is both significantly smaller and tracks the lecturer in a more natural way. Since we run our system as a post-process, we can analyse the entire video before making any physical changes to it. This allows our algorithms to analyse future events in advance and perform more robust edits to the video file. It is also possible to use more complex calculations at each stage of the pipeline, which should allow for fewer errors in lecturer-tracking.

Lectures are recorded using 4K cameras, which capture the whole view of the lecture area including any chalkboards (henceforth referred to as blackboards) as seen in Figure 1. The lecturer and blackboards (typically green or black) are then tracked by computer vision algorithms and framed by a small cropping window to simulate a virtual panning effect. This cropping window is set to a size smaller than the original 4K input, typically a 720p video frame; in effect, the system must determine how to move this window through the larger frame in a smooth and natural way which sensibly frames the lecturer and board

content. Once the cropped stream is generated, the output video can easily be streamed online due to its reduced file size.

The software has three main components (or stages); blackboard and usage detection, lecturer-tracking, and the Virtual Cinematographer (VC). Blackboard segmentation is the process of detecting where the blackboards are in the video and when they are used. The blackboard usage information allows the VC to make framing decisions. The tracking module has access to the full video and performs a lookahead to find and track the lecturer in the video. The person with the longest on-screen time is then selected as the lecturer. Finally, the VC uses all the information from the previous stages to crop the best possible view from the input video.

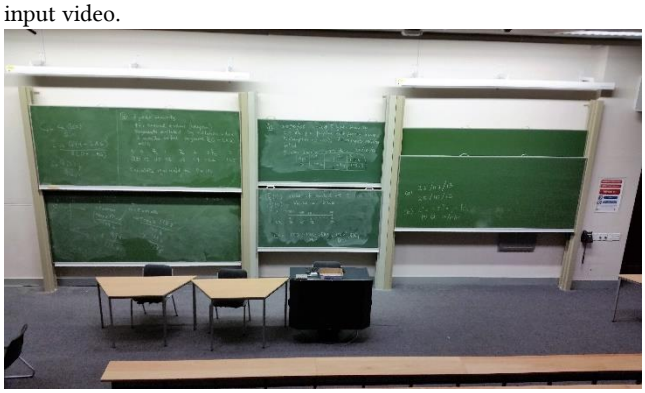

Figure 1 - Example lecture venue

Our results show a successful first pass implementation which can complete post-processing in under 3 times the input video's length. This is important because the high volume of lecture videos could result in a perpetual backlog if the processing takes too long. It is also worth noting that our implementation can reduce file size from 1.97 GB to 219 MB when using the MP4 CODEC (9 times smaller). The framing produced by the VC was also deemed to be visually pleasing by our expert evaluators and produced better results than some real-time tracking implementations.

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

2 BACKGROUND AND RELATED WORK

In this section, we provide a review of related work from the following fields: blackboard and usage detection, object tracking, and virtual cinematography. These fields collectively constitute a lecturer-tracking package.

2.1 Blackboard Detection

Blackboard and usage detection is performed in other approaches by using background modelling to separate the foreground from the background [6]. These approaches assume that the blackboard is static. Since the only motion is assumed to be the lecturer, the foreground motion can be classified as being the lecturer. These approaches use existing computer vision approaches such as background subtraction to build the functionality [7, 8]. Edge-detection algorithms can also be used to detect the features on the boards [9], which provide good results. Our work also uses an edge-detection approach for detecting the boards and uses feature detection for detecting when writing is added to it.

2.2 Tracking

Zhang et al. [10] present a tracking solution based on a pixel motion histogram that implements virtual panning. In addition to this, they also use a PTZ camera that can pan horizontally to track the lecturer when they move out of frame. They mention the performance issues with face detection regarding run time as well as robustness. To make their system robust to lighting changes and make it more efficient, the motion pixel histogram is only calculated around an area that surrounds the detected lecturer. This means that changes in lighting don't affect other areas of the screen as they simply aren't processed. While this solution appears to work well, their frame was a 640 x 480 resolution which is far smaller than the 4K frame we need to process. It is possible that performance would become a problem.

Arseneau et al. [11] present a tracking solution for classroom environments where the camera isn't mounted and pointed in an optimal position. A background subtraction technique that divides horizontal and vertical maxima into bins is used. The global maxima of these two axes are chosen as the centre point of a region of interest. These regions of interests are then processed using a 2:1 height:width ratio rectangle to output the location of the presenter. This approach is more robust to room setup but doesn't account for other movement in the scene, if two humans were to enter the view the region of interest could jump between successive frames.

2.3 Virtual Cinematography

Virtual Cinematography, also known as Virtual Videography and Computational Cinematography [12, 13], can generally be achieved by two different methods combined with various heuristics [12–17]. The first method involves taking existing frames and cropping them to a stream of a smaller size with a lower quality. The second method involves making the camera move autonomously (such as a Pan Tilt Zoom camera) in such a way that the resultant video mimics a video recorded by a human cinematographer.

There is a collection of seminal papers in the field of Virtual Videography and Computational Cinematography [7, 12, 13, 15] which suggest a system that can automatically create good quality video presentations from already recorded videos. These papers mention how important it is to understand what is happening in the video and to know whether it is beneficial to pan (or alter the current video stream in any way) before doing anything. The papers are an iterative improvement towards a functional system and, as the authors progressed, a list of principles and heuristics was derived over time. Notably, the system must attract the viewer's attention to what is important in the video; it must make effective use of both space and time by pacing the pan operations artfully; the viewer's visual interest should be kept since the size of the display is small and the attention of the viewer is limited. A successful VC should communicate this information to the viewer intuitively.

Rui et al. [3] describe their system as an automatic lecture tracking and recording system which is aimed towards recording lectures without a personal camera crew. The system tracks the lecturer, the members of the audience, the slides of the presentation and is also capable of selecting a stream from multiple camera outputs using a "Virtual Director" (VD) which controls a VC for each camera. This paper lists the main components necessary for an effective VC and concludes by noting that their implementation works almost as well as a manned camera but there is still potential for future work and improvements.

A VC should, therefore, accommodate the heuristics listed below when it evaluates how to generate output:

- The camera should focus on what is important.
- The VC should not frame the shots in any way that would confuse (or otherwise frustrate) the viewer but should guide their attention.
- There are advantages to post-production (such as using past and future information for the current frame and efficiency is not the most important concern).
- All pans need to consist of a single direction and should be considered as a separate operation to the other pans.
- The panning should be smooth, which means it should accelerate to a maximum speed and then decelerate towards the end of the pan.

3 METHODOLOGY

This section discusses the implementation of the system and has been divided into three sections, one for each part of the system. All the image processing methods are implemented using the OpenCV library [18]. The system was developed in C++ due to its performance capabilities [19]. The system is made up of three modules, described below.

3.1 Blackboard and Usage Detection

The board detection module is responsible for detecting the position and movement of the blackboards in the lecture venue and then determining when content is added or removed.

The scene is analysed by using edge detection to find all edges. These edges are then processed by an OpenCV algorithm *findContours* which is based on [20] and then outputs a list of points representing all the edges found. These contours are then bound by rectangles allowing us to find all rectangular shapes. This stage produces many rectangles that do not enclose the boards and we need to select only the rectangles that enclose boards. This is done by evaluating the rectangles box and contained pixels on the following set of criteria: pixel brightness, aspect ratio, and size. Brightness is evaluated by first converting the image into a binary image. A threshold is chosen that ensures the blackboards map to black in the threshold image; this is easily done since they are darker. We use aspect ratio – defined as the ratio width:height – and size to remove unwanted rectangles. The width needs to be longer than the height with an aspect ratio close to 1.8:1. This aspect ratio was chosen since we found that most lecture boards had an average lower bound aspect ratio close to this value. With regards to size, small rectangles that have a width less than 400px (pixels) and a height less than 300px are discarded. This was chosen since the boards usually occupy a substantial portion of the screen and small rectangles were not likely to represent boards. We found that this worked well.

Once all rectangles in the frame have been classified, the rectangles (containing blackboards) are then passed into a feature detection function. We require a way of detecting when the boards are being utilised. By using a feature detection algorithm, we can count the features in each rectangle and whenever we detect an increase in the number of features, we can assume that the board is being used. The Speeded Up Robust Features (SURF) [21] function was used to count the number of key points in each rectangle. An example of these features can be seen in Figure 2. SURF is robust and produces usable features regardless of handwriting style since it looks for corners or edges which are present in any style of handwriting.

Our primary goal is to locate moving boards since we expect these to more strongly influence VC decisions. Since these are often arranged above one another, we chose to group them in columns and to s count the number of detected features from the individual boards in each respective column. The feature count is evaluated and when an increase of features beyond a threshold occurred in this column it is flagged as being in use. An example of this can be seen in Figure 3. As seen in Figure 2, the feature detector also detects features of the lecturer and to prevent the system from classifying this as board usage, the threshold value was empirically chosen to limit the impact of these spurious detections.

Figure 2 - Features detected by SURF

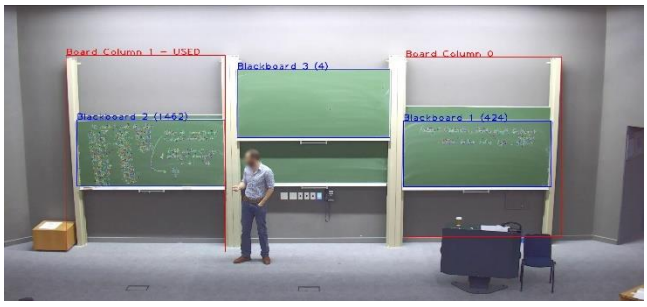

Figure 3 - Features tallied by column and usage is flagged

All this information is then passed onto the VC to make better framing decisions.

3.2 Movement Recognition and Tracking

We made use of a background subtraction [22] technique for our movement detection algorithm. We found background subtraction is cheap computationally, but still successfully detects movement. We interpret the detected movement by making assumptions about general lecture environments as well as human movement. Specifically, we used an absolute difference algorithm [23] subsequently filtered through a thresholding function. This provided mostly clean background subtraction with very little noise because of the very tight limits we chose for our thresholding function.

Next, we perform a morphological dilation [23] on the frame using a 3 x 3 rectangle structuring element. Subsequently, we apply a normalised box filter blur [23] using a large 15 x 15 filter kernel to make contours more recognisable. This minimises noise while making detectable edges more pronounced. After this, we find the contours of our edited frame the output of which is shown in Figure 4 below.

We noticed that contour chains with few nodes were often created by small differences between frames related to noise, light changes and camera refocusing. We, therefore, cull these; Human movements tend to have larger chains, as seen in Figure 4.

Figure 4 - Output after finding contours (left) and a processed bounding rectangle (right)

We generate a bounding rectangle around each contour chain as this makes it computationally cheaper to work with. This also simplifies decisions around linking or extending chains. We assume that lecturers will lecture above a minimum and below a maximum height as cameras will be pointed to the centre of the lecturing space. Given this assumption, we culled rectangles entirely outside this "lecturing band".

We also assess all rectangles' *width:height* ratio which if too wide are also culled. This is intended to cull tracked boards as shown in Figure 5.

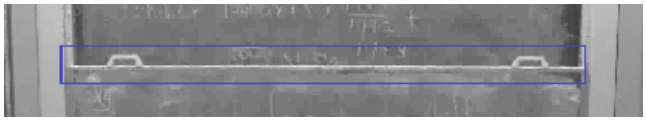

Figure 5 - Movement of board's top being detected

Once we have culled all logical oddities we needed to merge nearby and overlapping rectangles into a larger rectangle. By doing this, multiple tracked rectangles which are part of a larger single object, such as a person, should be correctly merged together. Figure 4 (right) shows the outcome of this clustering algorithm.

Two frame information redundancy checks are then performed. The first checks if the frame is too cluttered with rectangles. This can happen when too much motion is detected, including drastic light changes, or when the camera refocuses. In this case, it is difficult to extract any useful information. The second case is if no bounding rectangles are found. In both cases, a default rectangle is placed in the centre of the frame. This step is important otherwise frames are effectively discarded. The virtual cinematographer depends on complete information to avoid lagging the actual video.

Rectangles are tracked and associated with frames using our "Ghost" class and logic. This records how long a sequence of related rectangles has existed across frames and allows us to establish some form of persistence. The ghost tracks the movement detection rectangles across multiple frames recording how long an object is tracked as the number of frames it is tracked in. A new ghost is instantiated when a rectangle is found that doesn't intersect with any ghosts from the previous frame. This new ghost will have the dimensions of that rectangle as well as on screen time of 1 frame. In successive frames, if this new ghost intersects other rectangles, it will grow towards that intersecting rectangle's extremities as shown in Figure 6. Specifically, we relate each ghost corner to a nearby rectangle corner, such that each corner has only one other related corner. We then move each ghost corner to a position 75% of the distance between the ghost point and the rectangle point. We found this value helped the ghost retain a moderate size while still translating with the lecturer. It wouldn't scale too fast with large rectangle changes, such as a board moving, but also wouldn't shrink inwards too much if it intersected a smaller rectangle. This meant that the ghost was generally close to the rectangle's dimensions but allowed for variation in terms of size and placement.

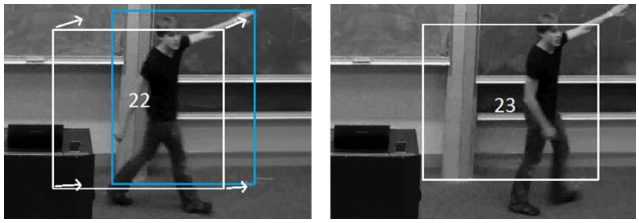

Figure 6 - A ghost (white) updating position towards the movement detected (blue)

If no rectangle intersection is found or a rectangle's intersection with the ghost is below a threshold percentage of the ghost's area, then the ghost will shrink inwards, reflecting uncertainty around the tracking result. Once the ghost has become small enough it will not be recorded in subsequent frames. This assumes that people in view will constantly be moving at least a small amount.

This should ensure that animate objects, such as people, are constantly tracked whereas inanimate objects such as boards are only tracked when they are moved. The resizing of the ghost based on intersecting rectangles allows the ghost to track movement laterally as in Figure 6. This assumes that people do not move quickly enough to exit the ghost between frames.

To handle occlusion of two or more tracked objects, merge and split algorithms were introduced. This means the highest screen time count of the merged ghosts is recorded and kept. Figure 7 illustrates the split algorithm running. It shows two rectangles within a ghost being a certain distance apart, greater than an empirically determined threshold, and therefore is split into individual ghosts.

As a form of soft reset in the case that the system picks up the wrong object as the lecturer, each ghost's time is reduced by two-thirds every 120 frames. We found through testing that 120 frames were frequent enough to affect any object visible on screen for more than 4 seconds. The choice of a 4-second window is based on the time it took to briskly walk across the 4K camera view.

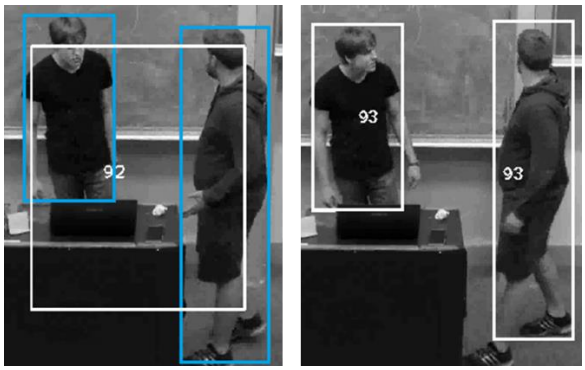

Figure 7 - Ghost (white) being split into two because of rectangle (blue) distance

$$valx = onScreenTime * \left(1 - \frac{xFromCentre}{centreX}\right)^2$$
$$valy = onScreenTime * \left(1 - \frac{yFromCentre}{centreY}\right)^2$$

Code Block 1 - Positional Importance Formula

This function (Code Block 1) continues the assumption that a lecturer will favour the middle of the view on the x and y-axis. Finally, the locations of the lecturer are saved as a vector of rectangles and can be accessed by the virtual cinematography module to perform the cinematographic logic. This will be discussed in the following section.

3.3 Virtual Cinematographer

The Virtual Cinematographer (VC) is split into 3 components: video lookahead and pan analysis, noise reduction and smooth pan sequencing, and final output.

The video lookahead component deals with identifying the changes in the lecturer's position. The VC has all the lecturer's positions and evaluates them one at a time. Each position is compared with its predecessor to determine a direction. All frames where the lecturer is moving in the same direction are accumulated into a pan operation. A pan operation is a collection of frames with a start and end defined in the 4K frame's coordinate space. The accumulation is stopped when the direction reverses. This process continues for all recorded frames and the collection of pan operations are passed onto the next component of the VC.

The next component evaluates the newly formed pan operations and determines which are too short to produce a meaningful pan. These short operations are combined with other short pan operations to make a *noise segment*. The noise segments are parts of the video where the cropping window does not move. This is done to remove the jitter from the cropping window's movements. As noise segments appear, they are recalibrated: the end position of the pan operation before the noise segment is moved to the start position of the pan operation immediately after the noise segment. We do this so that the cropping window does not jump from one position to another after pausing over a noise segment.

Once the pan operations are fully refined the last component writes the cropping window's enclosed area (for each frame) to the output file. The resultant video file has a resolution of 720p and resembles the motions a human camera operator would have made (instead of just a static camera). A visual example of this cropping process can be seen in Figure 8.

Figure 8 - Cropping window output as 720P

4 RESULTS

We ran the experiment on a laptop with the following specifications:

CPU: Intel Core i7-6700HQ @ 2.60 – 3.5 GHz
RAM: 16GB DDR4 2133MHz
Hard Drive: 7200RPM 1TB HDD
OS: Windows 10

We tested blackboard detection and the tracking modules individually to determine the robustness and speed of these parts of the pipeline – see Table 6 for the runtime results. To evaluate the quality of the VC, we ran the video processing pipeline with a full 4K lecture video as input to produce a complete 720p version. We sent the output video for expert evaluation to assess the quality of the video from a professional perspective. Since this is a time-consuming task the expert was unable to commit to more evaluation time and only one video was professionally assessed.

4.1 Blackboard and Usage Detection

Since the blackboard and usage detection was primarily being evaluated for feasibility, less time was devoted to testing this component.

The evaluation was conducted by manually reviewing a 50-minute lecture and flagging the frame timestamps at which each of the cases in Table 1 occurred. This was considered the control. This video was then passed through the blackboard and usage detection module and was compared to the control, by counting the number of times our system correctly flagged the corresponding case at that timestamp. Since this was a normal lecture, there was no set number of instances for each case and the results in Table 2 show a percentage of the experiment outcome successfully meeting the expected outcome of each test case. The six test cases used along with their expected outcomes can be found in Table 1.

Table 1 - Board usage detection test cases

No.	Test Case Description	Expected Outcome
1	Lecturer walks passes board without writing on it	Board not used
2	Lecturer gestures in front of the board (Hand movements, pointing, but not writing)	Board not used
3	Lecturer writes on board	Board used
4	Lecturer moves board with writing on	Board not used
5	Lecturer moves empty board	Board not used
6	Lecturer erases board	Board used

Table 2 - Results of board usage detection

No.	Test Case Description	Success (%)
1	Lecturer walks passes board without writing on it	100
2	Lecturer gestures in front of the board (Hand movements, pointing, but not writing)	100
3	Lecturer writes on board	83.8
4	Lecturer moves board with writing on	100
5	Lecturer moves empty board	100
6	Lecturer erases board	0

For test cases 1, 2, 4 and 5 the system matched the expected outcomes for each respective instance evaluated. In case 3, the system only correctly identified 83.8% of these cases as identified in the control. Finally, test case 6 failed entirely since none of the board erasing events triggered as usage. This is likely due to chalk dust being detected by the feature detector and hence the decrease in features not exceeding the threshold necessary to be classified as usage.

4.2 Movement Recognition and Tracking

The lecturer-tracking worked successfully for all normal cases of lecturer movement and only had difficulty in tracking abnormal behaviour such as quick running, multiple students crossing the room constantly, and continuous sharp light changes. We found that normal behaviour such as board movement, a single student crossing the field of view, and lecturer pacing had a successful tracking rate of 90%.

To test this module, we devised a set of 17 use cases shown in Table 4. The use cases cover how a lecturer might move in a real lecture. We then recorded lectures we set up specifically to test these 17 use cases with a 4K camera. These videos were then processed by this module. We analysed a set of 5 videos of real lectures. For each of our 17 use cases, we estimated a score (1 – 5) explained in Table 3. Note that we do not use likelihood in the regular statistical sense, but in a more colloquial way as explained in the Table.

Table 3 - Explanation of lecture use case scores

Likelihood	No.	Occurrence
Very Unlikely	1	Once in 5 videos
Unlikely	2	At least once in 3 videos
Possibly	3	Once in all 5 videos
Likely	4	2 – 5 times in all 5 videos
Very Likely	5	More than 5 times in all videos

Table 4 - Lecturer use cases

No.	Lecturer movement description	Likelihood (1 - 5)
1	Light movement, no pacing or gesturing.	5
2	Moderate movement, pacing and gesturing.	4
3	High movement, heavy pacing and gesturing.	4
4	Sudden and reoccurring light changes, moderate lecturer movement.	2
5	Moving boards often and light movement.	3
6	Projector screens rolling up and down alongside heavy movement and pacing.	2
7	Projector screens rolling up and down while stationary.	2
8	Move outside view and back in.	3
9	Stationary lecturing while student crosses.	3
10	Lecturer and student moving together, lecturer stops in the centre of view.	2
11	Student and lecturer approaching from opposite sides of view.	1
12	Lecturer runs across the view.	1
13	Lecturer plays “catch” with a student.	1
14	3 students giving a presentation.	1
15	Two students cross from either side of the room with the lecturer in the centre.	1
16	Students moving along chairs in the bottom of the view.	3
17	No movement with no one in the view.	2

The division between likely and unlikely use cases ultimately helped us evaluate the effectiveness of the lecturer-tracking module.

To evaluate the performance of each use case we stepped through the videos at a rate of 4 frames per step counting the number of times the program incorrectly identified the lecturer. Incorrect identification was counted when the tracking rectangle tracked the wrong person, a moving object or lagged the movement of the lecturer (having the lecturer completely outside of the rectangle but still tracking in their direction). In our results, we simplified the data by rounding frames to the nearest second. Using this result we evaluated the percentage of the use case that was correctly tracked. Our goal for success in a use case was 90% which represents almost the whole use case correctly tracked. The 10% margin of error means that if the lecturer is miss-tracked it would not be long enough to cause the viewer to lose any visual context.

We also evaluated the efficiency of each run, where processing time of more than 2 times the length of the video constitutes failure. This limit was chosen because the total processing time available to the system was limited to 3 times the length of the video to make it tractable.

Table 5 - Lecturer use cases results

No.	Likeli-hood	Video Length (s)	Process Time (s)	% Process Time	Did not track (s)	% Correct Track
1	3-5	57	66.192	116.13%	2	96.49%
2	3-5	31	32.756	105.66%	0	100%
3	3-5	48	46.648	97.18%	0	100%
4	1-2	85	90.575	106.56%	60	29.41%
5	3-5	40	44.879	112.20%	3	92.50%
6	1-2	48	56.975	118.70%	1	97.92%
7	1-2	46	52.147	113.36%	3	93.48%
8	3-5	40	45.824	114.56%	3	92.50%
9	3-5	72	85.613	118.91%	4	94.44%
10	1-2	17	19.67	115.71%	1	94.12%
11	1-2	16	21.707	135.67%	0	100%
12	1-2	24	28.274	117.81%	14	41.67%
13	1-2	32	40.115	125.36%	2	93.75%
14	1-2	162	156.31	96.49%	62	61.73%
15	1-2	30	37.577	125.26%	3	90%
16	3-5	32	36.397	113.74%	1	96.88%
17	1-2	77	76.248	99.02%	0	100%

From it's clear that the runtime of each use case passed the efficiency test. This can be ascribed to our approach of choosing simple, cheap algorithms such as background subtraction and the use of rectangle reasoning to track the lecturer. This is particularly important since a lot of processing time went to reading and decoding the large 4K frames where we couldn't find any efficiency gains. This I/O overhead represents a fundamental lower bound on how fast any processing can occur

The results in Table 5 show that most of the tracking tests pass with all the likely cases succeeding. We found that for most cases enough lecturer movement was happening for the system to find something large enough to track.

The tracking solution registers large movement well. This is because the first step of our algorithm employs absolute difference background subtraction. It also works well for moderate movement because of the rectangle clustering algorithm. Unfortunately, with very small or no movement there isn't enough information for our system to track the lecturer so we default to the centre of the screen.

The results show that our tracking fails the 90% test for 3 of the use cases. These are all unlikely use cases meaning that this shouldn't invalidate the usefulness of the tracking module.

Use case 4 evaluates many sudden light changes which caused the 4K camera (which attempts to do light correction on the fly) to reduce its effective frame rate and stutter. The test itself leveraged lots of sudden light changes in quick succession which is abnormal for lecturing conditions where the lecturer is more likely to make a single lighting change. So, while this test performed badly it represents a very unlikely sequence of events.

Use case 12 evaluates a lecturer running across the lecturing area. When processed, the lecturer is moving quickly enough to exit his or her ghost between frames. This means the ghost's position isn't updated towards the new rectangle position as a locational correlation wasn't found. This problem is a direct result of our frame sampling rate; if we skip fewer frames between a detection step this effect can be lessened or removed entirely, however, this would increase processing time for this module.

Use case 14 evaluates multiple students giving a presentation. This use case only tracked the lecturer correctly 61,7% of the time. Fundamentally this is meant to be a difficult use case for the module to handle. When 3 students are presenting there is no indication other than voice and nuanced movement to distinguish who the presenter is at any one moment. While our system was developed to handle temporary passing and occlusion of students it still only tracks one lecturer. With this in mind, the problems we noticed were the students who weren't lecturing but were in the view continued moving and thus retained their screen time count. Additionally, because the role of speaker is passed between students, the screen time counts are all mixed together. Therefore, the lecturer is often decided by who of the 3 is most central in the view.

The lecturer-tracking worked for all normal cases of lecturer movement and only has difficulty in tracking abnormal behaviour such as quick running, multiple students lecturing and continuous sharp light changes. Given these encouraging results, we believe this module could realistically be used for lecturer-tracking in real lectures.

4.3 Virtual Cinematographer

We sent a reference output video from the VC module for professional evaluation and we received the following remarks concerning the video quality:

- The video starts untidily due to the lecturer-tracking module since it struggles to identify the lecturer fast enough to be unnoticeable.

- The camera acceleration and deceleration are very smooth and effective at framing the presenter appropriately (in most cases). There is also no point at which the movement of the camera is jarring or unexpected.
- The resolution of the output video stream has more than enough clarity and detail for viewers to read the writing on the boards and see the lecturer's gestures (which are the main objectives for the VC to address). The camera height was slightly too low for some of the writing, however.
- The output video was compared to the output videos from competing approaches and the output of this program is significantly better than that of the Axis Digital Autotracking app which ran directly off the camera. It is also equivalent to (or even slightly better than) the Axis 5915 camera and LectureSight real-time tracking solution.
- The VC has some shortcomings in its framing of the lecturer when the lecturer is writing on the board.

4.4 System Runtime and Efficiency

Regarding runtime, the processing time taken to process a 50-minute length video, categorised by the operation can be seen in. A separate column shows the amount of time used for file Input and Output (IO) operations. The I/O time can be mitigated by using fast SSD, for example. Note that these results do not use parallel implementations of algorithms or GPUs, so there is certainly scope for improvement.

Table 6 - Runtime results

Operation	Time taken (Algorithm)	Time taken (File IO)	Total
Blackboard and Usage Detection	15m20s	27m15s	42m35s
Movement Recognition and Tracking	29m15s	27m15s	56m30s
Virtual Cinematographer	10m45s	32m15s	43m0s
TOTAL	**55m20s**	**86m45s**	**142m5s**

4.5 Discussion

Preliminary testing of the Blackboard and Usage Detection module showed that using a feature detection algorithm is feasible for this purpose and that further studies can be carried out in this field. The feature detection approach worked and could recognise the presence of handwriting. This is because feature detection looks for edges and corner key points and any handwriting style will always have these present.

The feature count threshold was chosen based on a small sample of lecture videos. This could also be an influencing factor in the failure of the board erasing test case and a more complex approach may be required to deal with handwriting detection artefacts.

The movement recognition and tracking module functions well but there is room for improvement. The movement recognition implemented with background subtraction techniques fundamentally lacks any context about the movement it picks up. While we have accommodated many contexts with our rectangle clustering and other checks, there are still other possible extensions such as utilising the colour characteristics of objects to help differentiate them. Such characteristics could also be used in the ghost reasoning section which struggled on tests where the identity of who was lecturing was unclear.

Based on initial testing, the VC produces acceptable framing but the cropping window is, at times, too low for some of the content on the higher boards. This problem arises from the exclusive use of horizontal panning to move the cropping window. When the lecturer is writing on the board, the VC seems to be worse at framing both. Improvements can thus be made to the heuristics involving the lecturer and the use of the boards.

For the proposed processing r pipeline to be tractable, we required that the time required for all video processing is within 300% of the input video's length. Our results show that video post-processing completed in less than 285% of the input video length, within the prescribed 300% threshold. Furthermore, we found that file I/O amounted to about 60% of the total system runtime. For the VC, the time spent on I/O, about 75%, was much higher since it also needed to save the smaller output video stream. The system reduced the input file size from 1.97 GB to 219 MB in the output video when using the MP4 CODEC - a 9-fold size reduction.

5 CONCLUSIONS

We have developed an automated post-production video editing system capable of reducing the resolution of the input 4K video by carefully selecting a small output cropping window to track the lecturer and local context through the larger 4K frames. The cropping window is moved in a way that mimics, to a large degree, the camera control decisions that a human camera operator would make when trying to keep the lecturer and board context in-frame using a camera with a smaller field of view.

The first system module, board detection and usage, shows promising initial results based on a feature detection algorithm, although excessive smudging from chalkboard erasure is problematic. The tracking module produces good results for the general use case and works in some unlikely use cases too. The VC reduces the video from 1.97 GB to 219 MB when using the MP4 CODEC, and the 720p video produced by the software is clear to the point where the board contents are clearly legible and lecturer's gestures are clearly visible - a major objective of the system. The system also successfully makes the cropping window move as smoothly as a human cinematographer's movements in a way that respects accepted cinematography best practices throughout the video.

6 FUTURE WORK

While expert opinion rates the video output as good, there is room for improvement in several aspects of the system. These are being actively pursued in ongoing research.

The writing detection in the blackboard and usage detection module can be extended to be more robust by using a larger sample of lecture videos using more complex heuristics to reject spurious clusters of feature key points.

The lecturer-tracking solution currently functions on the assumption that the lecturer will spend most of his or her time near the middle of the field of view. Certain lecture theatres may change this assumption. We could make use of the movement mask calculated in the pre-processing step to build a more robust solution and build this change into our algorithm (Code Block 1)

The framing section of the VC module has limitations and could be refined. The crop window currently only moves along the x-axis at a fixed position on the y-axis. This can be improved by adding functionality to tilt the cropping window. Zoom functionality can also be added in future version.

Currently, each module reads the entire file as it processes the video This makes the program run for longer than necessary since the frames being read are identical. This will be addressed in future work by reading the file in once and sharing this information across all 3 modules.

REFERENCES

[1] Demetriadis, S. and Pombortsis, A. 2007. e-Lectures for Flexible Learning: a Study on Their Learning Efficiency. Educational Technology & Society. 10, 2 (2007), 147–157.

[2] Gonzalez-Agulla, E., Alba-Castro, J.L., Canto, H. and Goyanes, V. 2013. GaliTracker: Real-Time Lecturer-Tracking for Lecture Capturing. 2013 IEEE International Symposium on Multimedia (Anaheim, 2013), 462–467.

[3] Rui, Y., He, L., Gupta, A. and Liu, Q. 2001. Building an intelligent camera management system. Proceedings of the ninth ACM international conference on Multimedia - MULTIMEDIA '01 (Ottawa, 2001), 2–11.

[4] Lampi, F., Kopf, S., Benz, M. and Effelsberg, W. 2007. An automatic cameraman in a lecture recording system. Proceedings of the international workshop on Educational multimedia and multimedia education - Emme '07 (Augsburg, 2007), 11–18.

[5] Friedland, G. and Rojas, R. 2007. Anthropocentric Video Segmentation for Lecture Webcasts. EURASIP Journal on Image and Video Processing. 2008, 1 (2007), 1–10.

[6] Wang, H. and Nguang, S.K. 2014. Video target tracking based on fusion state estimation. 2014 International Symposium on Technology Management and Emerging Technologies (Bandung, May 2014), 337–343.

[7] Wallick, M., Heck, R. and Gleicher, M. 2005. Marker and chalkboard regions. Proceedings of Mirage 2005 (Rocquencourt, 2005), 223–228.

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

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

[10] Zhang, C., Rui, Y., He, L.W. and Wallick, M. 2005. Hybrid speaker tracking in an automated lecture room. IEEE International Conference on Multimedia and Expo, ICME 2005 (Amsterdam, 2005), 81–84.

[11] Arseneau, S. and Cooperstock, J.R. 1999. Presenter tracking in a classroom environment. Industrial Electronics Society, 1999. IECON '99 Proceedings. The 25th Annual Conference of the IEEE (San Jose, 1999), 145–148.

[12] Gleicher, M.L., Heck, R.M. and Wallick, M.N. 2002. A framework for virtual videography. Proceedings of the 2nd international symposium on Smart graphics - SMARTGRAPH '02 (Hawthorne, 2002), 9–16.

[13] Heck, R., Wallick, M. and Gleicher, M. 2007. Virtual videography. ACM Transactions on Multimedia Computing, Communications, and Applications. 3, 1 (2007).

[14] Burelli, P. 2013. Virtual Cinematography in Games: Investigating the Impact on Player Experience. International Conference On The Foundations Of Digital Games (Chania, 2013), 134–141.

[15] Gleicher, M. and Masanz, J. 2000. Towards virtual videography (poster session). Proceedings of the eighth ACM international conference on Multimedia (Marina del Rey, 2000), 375–378.

[16] Jones, N. 2013. Quantification and Substitution: The Abstract Space of Virtual Cinematography. Animation. 8, 3 (2013), 253–266.

[17] Nagai, T., Toyota, T., Nagoya, T., Nishizawa, K. and Imai, M. 2013. Implementation of high-definition lecture recording system for daily use. IEEE Global Engineering Education Conference (Berlin, 2013), 520–525.

[18] OpenCV library: http://opencv.org/. Accessed: 2017-05-17.

[19] Prechelt, L. 2000. An empirical comparison of C, C++, Java, Perl, Python, Rexx, and Tcl for a search/string-processing program. (2000).

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

[21] Bay, H., Ess, A., Tuytelaars, T. and Van Gool, L. 2008. Speeded-Up Robust Features (SURF). Computer Vision and Image Understanding. 110, 3 (2008), 346–359.

[22] Ponce, J. and Forsyth, D. 2012. Computer vision: a modern approach. Prentice Hall.

[23] Thompson, M., Gonzalez, R.C.R., Wintz, P., Woods, R.E.R. and Masters, B.R. 2002. Digital image processing.