Virtual Cinematography and Tracking Heuristics for Presenter Tracking

Mohamed Tanweer Khatieb
Department of Computer Science
University of CapeTown
Rondebosch,7701, South Africa
khtmoh003@myuct.ac.za

ABSTRACT

The increase in camera quality and reduction in price has yielded an opportunity for Universities to install cameras in the lecture venues and record the lecturers' presentations for students to review at a later stage. This paper explores the reduction of the 4K video recordings to a bandwidth conserving 720p stream using cinematographic heuristics and rules to guide the extraction of a smaller window from the larger video frames. This small window effectively creates a region which can be panned across each high-resolution 4K frame, as if moved by a "virtual cinematographer". The camera panning in the output video should, as closely as possible, resemble the panning that a human cameraman would produce.

Our results were evaluated by experts at the Centre for Innovation in Learning and Teaching (CILT). In their evaluation, they stated that The Virtual Cinematographer module produces virtual camera motions that match a human cameraman and even surpasses existing proprietary systems.

There is still much room for improvement and therefore future work is necessary before this system can be deployed.

CCS Concepts

• **Computing methodologies →Computer vision problems**

• **Computing methodologies →Video segmentation**

• **Computing methodologies → Tracking**

Keywords

Virtual Cinematographer; 4K video; Panning; Scope; Tracking

1. INTRODUCTION

The rise in imaging technology has brought with it the increase in camera quality [3]. The new 4K resolution is of size 3840 x 2160, twice as detailed as a 1920 x 1080 High Definition (HD) resolution image. With this improved resolution comes the potential to improve the quality of lecture recordings that are currently offered to students in many universities. The University of Cape Town (UCT), in conjunction with the Centre for Innovation in Learning and Teaching (CILT), is currently considering the use of wide-angled 4K cameras to capture lectures where the entire presentation space is always in view. This form of video, however, is far too large to stream over the internet and so a smaller video file needs to be generated. These recordings are also not necessarily processed sufficiently once they are recorded to make them pleasing to the viewer [3-5] and it is practically and financially infeasible to hire a dedicated filming crew for each lecture venue [10]. This project explores the downsizing of the 4K video stream by cropping each frame down to a 1280 x 720-pixel resolution format, also known as 720p. This is done in a way that makes the resultant video appear to have been recorded by a human cinematographer using a technique known as Virtual Cinematography [3-5].

The project consists of three parts: 1) blackboard segmentation, which removes the blackboard space into a separate stream for viewers to get a view of all the boards in one view; 2) lecturer tracking, which identifies the possible lecturers and follows these candidates based on their times on screen and proximities to the boards, and 3) the Virtual Cinematographer (VC) which makes use of the tracking section and produces a collection of panning operations. This last component is the subject of this paper.

The VC operations are executed and the specific section of the 4K frame is cropped and saved to a new output file. The cropping window is effectively what is being panned.

The VC makes decisions such as deciding when to pan and how far, how the lecturer should be framed and how fast to react to the lecturer's movements. The VC is the last stage of the project pipeline since it relies on all the information from the previous sections. It needs to take the positions of the lecturer and determine whether it is necessary (and beneficial) to pan and then take appropriate action. The output of the VC is also the final output of the project which further justifies its place at the end of the pipeline.

The VC has three modules which it uses to produce the final output: the video look-ahead and pan analysis module, the noise reduction and smooth pan sequencing module and the final output module.

Video Look Ahead and Pan Analysis

This module looks at lecturer positions and determines how to pan the cropping frame.

Noise Reduction and Smooth Pan Sequencing

Once the initial pan operations are identified, this section investigates these operations and determines if they are effective pan operations or if they are simply noise (short, jittery motions) to be removed from the pan sequence. The revised pan sequence is then sent off to the final output section.

Final Output

This moves a 720p window along the pan operations and crops the area inside to the output video file. The result is a 720p video which seems to have been produced by a human cameraman.

The quality of the VC's motion is the most important factor when it comes to panning the cropping window over the 4K frames. It needs to frame the lecturer at all times and in the appropriate way for each situation. If the lecturer is referring to something on the right of the screen the lecturer should be framed on the left of the frame and the remainder of the frame should contain as much of what is being referred to as possible. This rule should apply for the mirror case as well. If the lecturer is not referring to anything the frame should be positioned in such a way that the lecturer is in the centre of the frame.

The primary research question for this paper is related to the quality of the output and the efficiency of producing this output:

How close can virtual cinematography come to matching a human cinematographer using virtual cinematography heuristics?

We found that the VC could match a human cinematographer for a basic implementation (such as a lecture venue) to the extent that a human cinematographer could barely tell the output was generated automatically.

The remainder of the paper is laid out as follows: Section 2 refers to the background of Virtual cinematography and its related work, Section 3 goes into the details of the design and implementation of the VC, Section 4 lists the results, discusses them and concludes.

2. BACKGROUND AND RELATED WORK

Virtual Cinematography, also known as Virtual Videography and Computational Cinematography [3, 5], can be achieved by two different means with the use of various heuristics [2-6, 9]. 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.

Gleicher et al. [4] suggest a system that can automatically create good quality video presentations from already recorded videos. In this paper they 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. This paper also suggests using special effects to enhance the viewers' experience without removing the information that is being presented. In the end they found that, while even the most basic implementation of such a system is capable of improving the quality of the video, there is still much potential (and need) for future work. This leaves scope for work on a VC in this module.

Rui et al. [10] 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 particular 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 which are: sensors and cameras for input, rules and heuristics to determine what the camera should do relative to the context and a strong communication between the VCs and the VD. The paper concluded by saying their implementation works almost as well as a manned camera but there is still potential for future work and improvements. This module focusses on the rules and heuristics mentioned above.

Gleicher et al. [3] suggest improvements for their earlier work in [4]. In this paper the improved system allows for the cheaper recording of lectures without needing a crew during or after recording. This paper lists the heuristics necessary for VC, namely: 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. The VC should (therefore) communicate the information to the viewer intuitively.

Heck et al. [5] build on their past work in 2002 [3] by creating a VC capable of post-processing videos using the rules and heuristics listed in aforementioned papers. The system uses stationary cameras to capture the videos. These videos are then processed with computer vision and signal processing methods to gather information about the scene. A planning algorithm is responsible for choosing the best shot to use (given the information from the scene) and finally an image synthesis method creates new images as output from the original video. This paper identifies four key factors required to create realistic VC results which are appreciated by users. They use syntactic cues, such as the contents of the boards and the lecturer's gestures and position, to help in deciding when (and how) to pan the camera. Post-production (offline) processing allows a look-ahead advantage to help the system know when the lecturer will move in such a way that will require panning or adjustment. This paper finds the evaluation of such a system to be highly subjective. Their results show that, while they obtained similar results to a human cinematographer, the output time for a ten-minute video segment took one hour and fifteen minutes.

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

The camera should focus on what is important. The panning should be smooth, which means it should accelerate to a maximum speed and then decelerate towards the end of the pan. All pans need to consist of a single direction and should be considered as a separate operation to the other pans. 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). The VC should not frame the shots in any way that would confuse (or otherwise annoy) the viewer, but should guide their attention.

We implement all of the aforementioned heuristics.

3. DESIGN AND IMPEMENTATION

3.1 System Constraints and Requirements

CILT has provided their system constraints and they are listed below:

The system must be able to run on Linux (Ubuntu). They use 24 CPU blade servers each with 96 GB of RAM. These servers run Virtual Machines (VMs) which simulate 4-8 CPU cores and 4-16GB of RAM per VM. These VMs also make thorough use of multi-threading techniques to optimise the output times for their videos. Their system would be better suited to an implementation without GPU optimization and the whole process should be no longer than 300% of the input video length.

The functional requirements for the virtual cinematographer are as follows: it needs to take in the lecturer's positions (in coordinates) and pan smoothly when necessary while cropping each 4K frame to an output video with a 720p resolution.

In terms of the non-functional requirements, the final video should be smooth and visually appealing to the viewer (see above constraints for more non-functional requirements).

3.2 System Overview

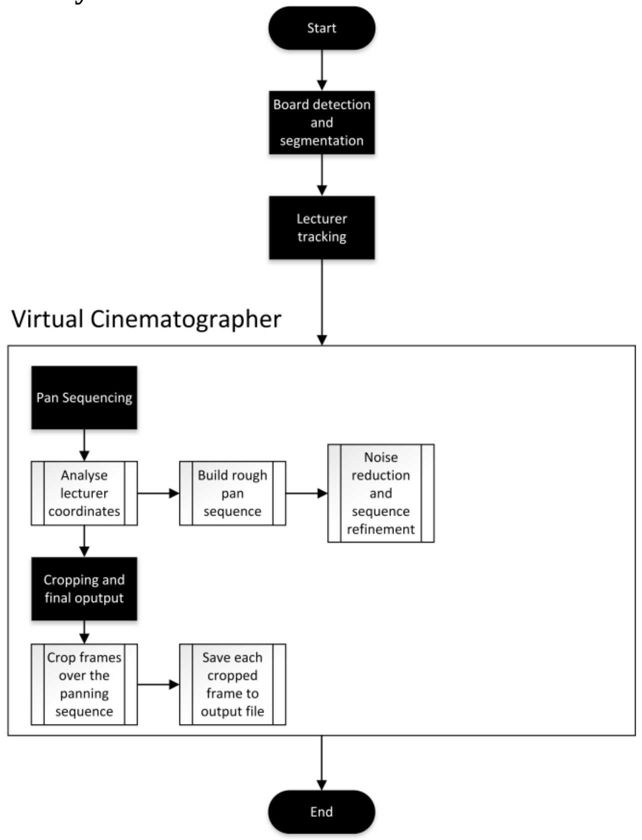

Start
Board detection and segmentation
Lecturer tracking
Virtual Cinematographer
Pan Sequencing
Analyse lecturer coordinates
Build rough pan sequence
Noise reduction and sequence refinement
Cropping and final oputput
Crop frames over the panning sequence
Save each cropped frame to output file
End

Figure 1: Virtual Cinematographer overview in relation to the whole system

The VC takes the lecturer positions from the lecturer tracking pipeline module and creates pan operations. These operations are then refined by removing noisy data to generate a smooth output free of jarring motions.

The output of this section in the pipeline is the final 720p output video.

3.3 Design

3.3.1 Video Look Ahead and Pan Analysis

This section looks at the positions of the lecturer (given by the tracking section) and processes them. The next position in the sequence is compared with the current position and the difference will determine which way the lecturer is going. This process is repeated (while keeping the current direction in storage) until the next position is no longer consistent with the current pattern (i.e. if each point was decreasing in the x-axis there is now suddenly an increase) which signifies a change in direction. This information is used to determine the start and end points of a pan operation. This pan operation is stored and the program continues to seek out the next pan operation in the same way. Once this is complete the system has a rough timeline for pan operations.

3.3.2 Noise Reduction and Smooth Pan Sequencing

This section reviews each of the pan operations. It checks the length of the pan and whether this length exceeds the width of the frame (i.e. whether the lecturer has left the current frame). If this is true, then the pan is kept as an operation. If this is false, then the operation is converted into a noise element and the next element is evaluated. Also, if this length is below a certain threshold, it combines it with the next (or previous) noise element. Once this process is complete, the system has a refined pan operation timeline.

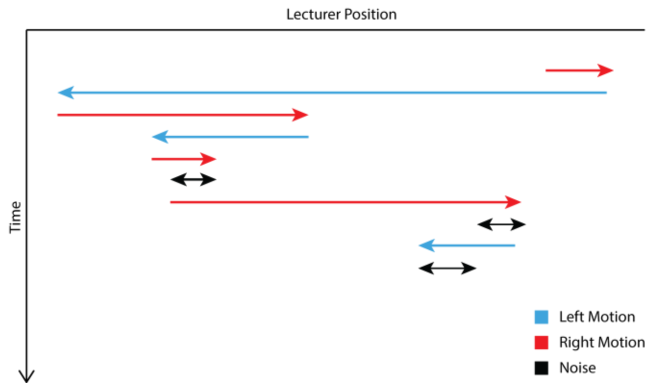


Figure 2: Refined pan operation timeline

3.3.3 Final Output

The cropping frame is made to pan over each pan operation while cropping each frame along the way (using the function: 1 + Cos (x - 135) restricted to a domain of 0 to 360 degrees for smooth motion) and writes these cropped sections to the final output video file. The cropping frame moves over the red lines (see Fig. 2) from left to right and over the blue lines from right to left. Black (noise) lines are not panned over. The cropping frame stays in its last position on these lines.

The image below shows how the pan is made smooth. The frame on the left is the start position and the frame on the right is the end position. The cosine curve is drawn above the pan to illustrate how the change in offset happens along the pan operation (short offsets which increase and then decrease in size).

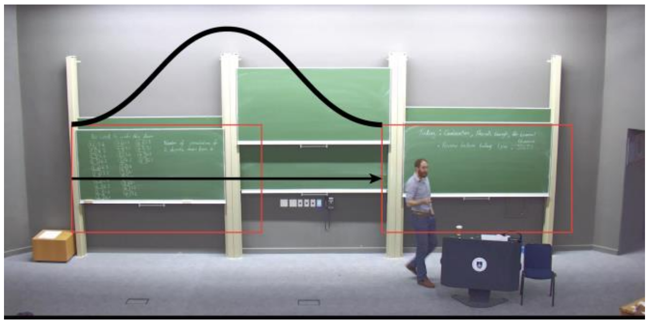


Figure 3: Using a cosine curve for smooth panning

3.4 Implementation

The program was written in C++ for efficiency purposes and we made thorough use of the OpenCV Library for the image processing algorithms because the solutions are open source, robust and efficient. We used CLion as our development environment.

3.4.1 Initialisation

The VC begins by reading a vector of bounding rectangles indicating the lecturer's position (passed through from the lecturer tracking module). The x-coordinates of the lecturer are taken into consideration while the position of the camera in the y-axis remains constant for a stability in the camera's panning. To find this fixed y-value, we iterated over all bounding rectangles and averaged the centre positions. We achieved better results by taking the average over all the rectangles (instead of a large sample of rectangles). This results in a lower variance of the final y-coordinate for the lecturer's centre. We needed to know the lecturer's centre to position the camera at an optimal height. Because the rectangles grow in size, depending on the lecturer's gestures and movements, we decided to make the centre of the bounding rectangle the position of the lecturer. These x-values were then stored in a vector and used to determine the pan operations for the cropping window.

3.4.2 Pan Operations

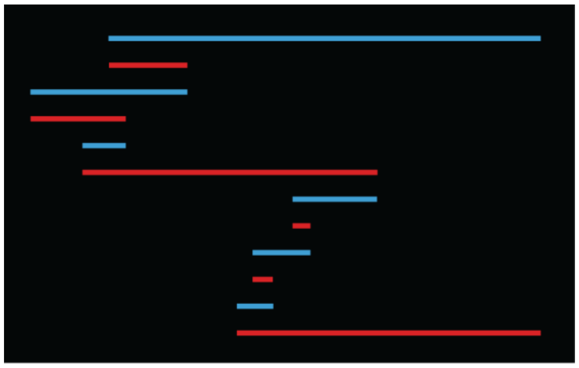
The next stage uses these coordinates to produce line segments representing the motion (and direction) of the lecturer (known as pan operations).

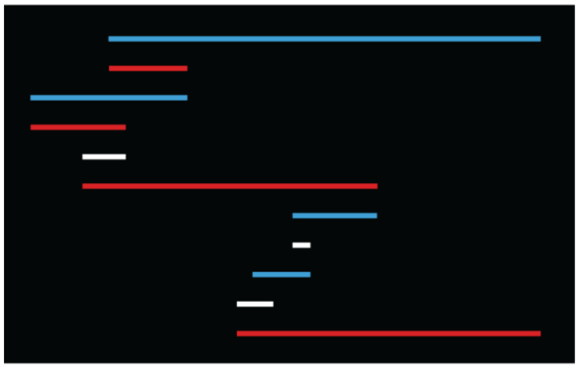
```
Counter = 0
for all lecturer coordinates:
    if previous x < current x AND direction is right
        counter++      // number of frames
    else if previous x > current x AND direction is left
        counter++      // number of frames
    else
        set start point as the old end point;
        set end point as current position;
        build pan operation;
        //using start point, end point and counter
        add pan operation to operation vector;
```

These pan operations are stored as objects and contain the start point, end point, number of frames and direction of the pan. This information will be needed when the cropping window is made to pan smoothly over these operations to produce the output video.

3.4.3 Cull Motion

This stage counteracts the jittering artefacts produced by the tracking system. Many small pan operations exist and produce very jarring pans. We reduce this and instead create one long line where possible, by making the cull motion stage take in a threshold parameter and remove all lines whose length is shorter than this threshold. These "noisy" lines are condensed into one pan operation which is later considered to be noise margins. These noise margins do not store direction, however they still store start and end points which are the extremes of all the short lines that were culled. These noise margins also store the sum of all the frames from the culled operations. It is very important that the total frame count (as represented by the pan operations) after processing is exactly the same as the number of frames in the input video to avoid skipping frames and producing camera jerk artefacts.


(a) Visualised output from pan operation section


(b) Visualised output from cull motion section

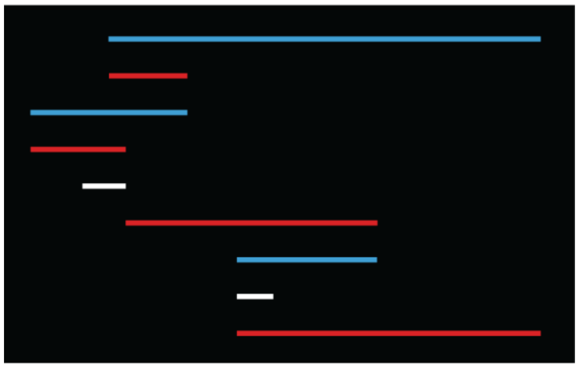

(c) Visualised output from repair culling section

Figure 4: Visualisations of the pan operations at various times during the VC module

One of the side-effects of the cull motion stage is that the end points of one line may (potentially) not line up with the start point of the next line when there is noise in between the two. This produces a jumping effect when panning over these operations. To repair this, a method iterates over all the pan operations and, when a noise margin is encountered, the next pan operation's starting point is updated to be the same as the end point of the operation before the noise. Later, during development, this method became obsolete as it often produced undesirable results such as panning past the lecturer's actual position. This functionality was built into the panning section instead and will be discussed in more detail there. This provided a smooth transition between the operations but an issue remained. Sometimes there was noise between pan operations that have the same direction. This caused the operation to split up, which didn't produce a smooth pan, especially if the noise was very short (i.e. the lecturer didn't stop walking). To work around this, a repair method was created to take in a frame count as a threshold. If a noise margin is encountered between two pan operations that move in the same direction and the number of frames in the noise margin is less than this threshold, these pan operations are stitched together to form a continuous pan operation. The frame count in the noise margin (between the now merged operations) is added to the frame count of the stitched pan operation. We found that, when the threshold was larger than 29 (i.e. one second when video is 29 frames per second), the pan was too slow in certain cases where the pan operation was very long. This resulted in the lecturer moving out of the frame for a short period until it has a chance to catch up again. Any noise that is less than 20 frames produced better panning results with a smoother output.

3.4.4 *Attaching board Usage*

The pan operation class accommodates for attaching board usage information (which is provided by the board segmentation module). The board usage is stored in a vector and provides the side of the venue on which the board was used (left or right) and the frame number at which it occurred. We iterate over the board usage vector and, for each board usage found, find the corresponding pan operation by its frame number. This operation is then set to contain the direction. We use this direction in the panning section to frame the board in use and the lecturer at the same time.

3.4.5 *Panning section*

The panning takes in a vector of refined pan operations which have been processed by the aforementioned stages. The panning section then iterates over these operations as follows:

```
//Loop over all motion lines
for each motion line:
  if not noise:
        pan (start point, end point, frame count, board
        usage, crop rectangle vector);
  else:
        for frame count:
            add last crop rectangle to crop rectangle vector;
```

The pan function calculates the amount by which the rectangle should be shifted on each frame to achieve the smooth panning effect. It uses a cosine function, cos (x - 135) + 1, to calculate the distance by which the cropping window should move for each frame. The board usage only adds an offset depending on the side of the venue. If, for example, the right board was used, we add the offset to the endpoint such that the pan includes more of the board and places the lecturer at the left of the cropping frame (and vice versa for the mirror case). Each pan iteration adds crop rectangles to the vector which will be used in the cropping stage.

To prevent panning over operations which do not leave the frame, a 20% bounding margin was added to both sides of the video stream. The pan operation length is evaluated and if it exceeds one of the bounding margins it will pan. When noise is encountered, the same crop rectangle is pushed back for the number of frames stored in the pan operations. If the noise intersects the margins and exceeds 1 second (or 29 frames), it will centralise the lecturer by panning.

3.4.6 *Cropping & Writing to File.*

In this stage the vector containing crop rectangles is iterated over and, for each rectangle, the corresponding frame is cropped and written to file. The cropping is done using OpenCV to set the Region Of Interest (ROI) to be the position of the corresponding crop rectangle for each input frame.

4. RESULTS

The experiment was run on a Gigabyte laptop with the following specifications:

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

RAM: 16GB DDR4 2133MHz

Hard Drive: Samsung 850 EVO SSD

OS: Windows 10

The program (i.e. all 3 modules) was run with a full lecture video as input and generated a full length 720p output video.

This video was then sent to the experts at CILT for a qualitative assessment of the VC.

4.1 Discussion

We timed the VC module and found that for a 50-minute input video the VC took an average of 43 minutes.

CILT has restricted our implementation (as a whole) to run within 300% of the input video's length. The VC takes less than 100% of the input video's length to complete and is therefore within the limits of the modular constraint (i.e. one third off the whole system's time).

Stephen Marquard [8], the coordinator of the Learning Technologies portfolio at CILT, made the following assessments of the VC module:

1. The video starts out untidily due to the lecturer tracking module since it struggles to identify the lecturer fast enough to be unnoticeable.
2. 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.
3. The resolution of the output video stream has more than enough clarity and detail for viewers to read the writing on the blackboards 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.
4. The output video was compared with the output videos from rival programs and the output of this program is significantly better than that of the Axis Digital Autotracking app [1] which can run directly off the camera. It is also equivalent to (or even slightly better than) the Axis 5915 camera and Lecturesight real-time tracking solution [7].
5. The VC has some shortcomings in its framing the lecturer when the lecturer is writing on the board.

The system was able to address the issue of not being able to find the lecturer in the beginning of the lecture by keeping the camera zoomed out and only zooming in once the lecturer has been identified confidently.

The experts at CILT are happy with the smoothness of the cropping window's panning. They believe the students watching these videos would be happy with the pan quality.

The video could benefit slightly from a height adjustment such that the lower boards are fully in view and some of the upper boards are also in the frame.

The experts at CILT suspect the blurriness which happens during some pans is a result of the physical camera optics and sensors instead of the processing.

The quality of the output is equivalent to (and possibly better than) some of the most prominent rival software systems which are freely available. This means our system can be used for post-production of lecture recording in the near future (possibly once the program is a little more refined).

Stephen asked Kristi Edwardes, a human cinematographer from CILT who has worked on Massive Online Open Course (MOOC) videos, what she thought of the program's output video with regards to the VC module. She said she was unable to tell that there was a Virtual Cinematographer doing the camera movements. The research question: *How close can virtual cinematography come to matching a human cinematographer using virtual cinematography heuristics?* Can therefore be answered with the following statement.

If a program is written with heuristics which focus on the main principles of cinematography, then such a VC could match a human cinematographer in quality (for simple purposes at least).

4.2 Conclusion

This project sought to create a video post-production program capable of outputting video files which could match the quality of a manned camera. It was born from necessity since 4K footage is far too big for students to stream online and it has achieved the goal of reducing the file size significantly [from 1.97 GB to 219 MB using the MP4 CODEC]. The system has produced output with a clear 720p resolution such that the content is easily visible and all board content is legible (this is a major objective for the system). We also succeeded in making the camera movements as smooth as a human cameraman's so that students viewing the output content would be comfortable throughout the duration of the video (this was the other primary objective).

The program is open source and freely available so CILT will be able to adopt the software and adapt it to their requirements.

4.2.1 Future Work

While expert opinion rated the video output as good, there are several aspects which require improvement.

The framing section of the VC module is not working fully and could use some refinement. It utilises board usage data to decide how the crop window should be positioned such that both lecturer and the content being referred to are displayed.

The crop window currently moves on a fixed y-coordinate but this y-value is too low and the upper parts of the boards are cut off. This should be adjusted in future iterations of the software.

The lecturer tracking section should be improved by adjusting the heuristics at the beginning of the program to be less aggressive when looking for a lecturer. The VC should be made to remain zoomed out until the lecturer is identified with an acceptable level of confidence.

5. REFERENCES

[1] Axis. Axis Digital Autotracking. 2016, 11/16 (2016). DOI=http://www.axis.com/za/en/products/axis-digital-autotracking.

[2] Burelli, P. Virtual cinematography in games: investigating the impact on player experience. Foundations of Digital Games, (2013).

[3] Gleicher, M. L., Heck, R. M. and Wallick, M. N. A framework for virtual videography. In Anonymous *Proceedings of the 2nd international symposium on Smart graphics*. ACM, 2002, 9-16.

[4] Gleicher, M. and Masanz, J. Towards virtual videography (poster session). In Anonymous *Proceedings of the eighth ACM international conference on Multimedia.* ACM, 2000, 375-378.

[5] Heck, R., Wallick, M. and Gleicher, M. Virtual videography. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 3, 1 (2007), 4.

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

[7] Markus Ketterl, D., Christopher Brooks, D., Florian Schimanke, M., Wulff, B., Fecke, A., Rupp, L. and Hamborg, K. LectureSight: an open source system for automatic camera control for lecture recordings. Interactive Technology and Smart Education, 11, 3 (2014), 184-200.

[8] Marquard, S. RE: Track4k - Evaluation of panning . (Nov, 16 2016).

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

[10] Rui, Y., He, L., Gupta, A. and Liu, Q. Building an intelligent camera management system. In Anonymous *Proceedings of the ninth ACM international conference on Multimedia.* ACM, 2001, 2-11.