EPiC Series in Computing

Volume 85, 2022, Pages 15–35

Proceedings of 43rd Conference of the South African Institute of Computer Scientists and Information Technologists

Low Resource, Post-processed Lecture Recording from 4K Video Streams

Charles Fitzhenry[1], Tanweer Khatieb[1], Patrick Marais[1], and Stephen Marquard[12]

[1] Dept. of Computer Science, University of Cape Town, Cape Town, South Africa
patrick@cs.uct.ac.za
[2] Centre for Innovation in Learning and Teaching (CILT).

Abstract

Many universities are using lecture recording technology to expand the reach of their teaching programs, and to continue instruction when face to face lectures are not possible. Increasingly, high-resolution 4K cameras are used, since they allow for easy reading of board/screen context. Unfortunately, while 4K cameras are now quite affordable, the back-end computing infrastructure to process and distribute a multitude of recorded 4K streams can be costly. Furthermore, the bandwidth requirements for a 4K stream are exorbitant - running to over 2GB for a 45-60 minute lecture. These factors mitigate against the use of such technology in a low-resource environment, and motivated our investigation into methods to reduce resource requirements for both the institution and students. We describe the design and implementation of a low resource 4K lecture recording solution, which addresses these problems through a computationally efficient video processing pipeline. The pipeline consists of a front-end, which segments presenter motion and writing/board surfaces from the stream and a back-end, which serves as a virtual cinematographer (VC), combining this contextual information to draw attention to the lecturer and relevant content. The bandwidth saving is realized by defining a smaller fixed-size, context-sensitive 'cropping window' and generating a new video from the crop regions. The front-end utilises computationally cheap temporal frame differencing at its core: this does not require expensive GPU hardware and also limits the memory required for processing. The VC receives a small set of motion/content bounding boxes and applies established framing heuristics to determine which region to extract from the full 4K frame. Performance results coupled to a user survey show that the system is fit for purpose: it is able to produce good presenter framing/context, over a range of challenging lecture venue layouts and lighting conditions within a time that is acceptable for lecture video processing.

Keywords: Presenter Tracking, Automated Lecture Recording Systems, Video Segmentation,4K video post-processing, Virtual Cameraman

1 Introduction

Teaching methods at universities and colleges are constantly evolving, as new technologies enable alternative ways of providing access to instruction. One area that has seen a slow but

steady advance is *lecture capture* — the recording of live lectures as an additional learning resource [7]. Once a lecture has been recorded it can be reviewed by students, in their own time, as a supplement to learning. Newer recordings are often based on very high definition 4K video recording, which provides exceptionally crisp images and the ability to zoom in on details (such as writing on a classroom board).

While this technology is convenient, it comes with additional overheads for both the student and the institution. For the institution, the lecture capture hardware and any necessary video processing infrastructure must be purchased and maintained, and this cost can be significant for large institutions with many recordings per day. For the student, the bandwidth required to download the videos — particularly full 4K definition videos — is an obstacle. A 4K recording of a 45-minute lecture can easily run to 2GB, so over the course of a lecture week the data required to download such content may not be affordable, especially when mobile data is used. These issues are exacerbated in developing countries, where university budgets and disposable income are often severely constrained. Indeed, this is the context within which this work was undertaken.

One solution, explored in detail in this work, is to utilise a *virtual cinematographer (VC)*, which defines a much smaller, fixed-size *cropping window*, within the 4K frame, to provide a context-centred view of the presenter. This smaller window should follow the presenter and provide sensible framing as they move about the venue, interact with boards and so on. Our results show that the output of this context-focused reduction results in a mean video file size reduction of 81% over the 4K stream.

The VC is driven by an image processing front end that analyses video frames to extract contextual information. In keeping with our need to reduce hardware costs for the institution, we limit ourselves to frame processing algorithms that are computationally cheap and can run on general-purpose commodity servers without requiring a Graphical Processing Unit (GPU). GPUs are typically expensive components and form an integral part of many modern deep-learning solutions in related domains. More specifically, we implement and evaluate a video processing pipeline which is based on frame differencing, on the front-end, and fast VC framing algorithms and heuristics on the back-end. The algorithms are designed to support high video throughput with a limited memory footprint, thus limiting the memory requirement for processing 4K video clips.

This project builds on earlier development work in this area, which led to a prototype, open source system *TRACK4K* [1]. The earlier prototype system suffered from tracking inconsistencies and only allows horizontal VC panning.

2 Background and Related Work

The design of a lecture capture system should aim to provide output videos that reduce the cognitive load of the learner and direct the viewer's attention to what is important [10]. *LectureSight* [24] is an open source live camera tracking system for lecture recording that requires expensive Pan-Tilt-Zoom (PTZ) cameras. In our context, live broadcasting is not required and the captured video can be post-processed and distributed. Post-processing also has the advantage of looking ahead in the video to make better tracking decisions [11] and is often used in conjunction with (one or more) high-resolution cameras to extract a smaller crop region from the input frames enclosing the presenter [4]. Our work follows this approach, although the framing decisions are not managed by the front-end tracking module, but deferred to the VC.

[1] https://github.com/LectureTracking/trackhd

Presenter Detection: The significant successes of deep learning (DL) and other machine learning approaches have led researchers to focus on exploring these state-of-the-art techniques [23]. Unfortunately, deep learning applications typically depend on GPU acceleration to obtain the full speed benefits for inference and learning. For example, in the pose estimation system presented by Cao et al. [2], inference time using the GPU is 36ms compared to 10396ms for the CPU-only version. While writing detection can be run infrequently (every nth frame, for small n), presenter detection must run much more often to cope with a moving presenter. This means a DL approach infeasible for our context, if used without a GPU.

In contrast, traditional motion detection approaches, do not require high performance hardware or GPU acceleration and are often used on embedded processors of IP cameras which have limited processing power [28].

To detect a presenter, an object detection scheme is typically used. Background subtraction, temporal/frame differencing and optical flow are three prominent techniques used to perform object detection [19].

Optical flow provides the best accuracy, but is computationally the most expensive. Temporal differencing is faster than background subtraction [19] and suitable for scenarios with changing backgrounds, which is required in our project since boards and screens can move.

In the approach of Mavlankar et al. [16], motion, skin colour and background subtraction is used to track the presenter across frames. Using a fixed skin colour is problematic due to varying lighting conditions and natural skin colour variability. Yokoi and Fujiyoshi [26] track the presenter using temporal differencing and thresholding, which is a useful approach in our context. The detected motion is clustered and the largest region is assumed to be the presenter. This assumption poses a problem in venues where boards or projector screens can move, since this will be detected as the largest motion. Furthermore, neither approach discusses the effects of students walking into view and how that might affect the tracker.

Writing Detection: Writing detection in the context of this project includes detecting writing on screen and blackboard surfaces, and is aimed at providing additional contextual information to assist and improve the framing decisions of the VC. Segmentation by colour [12], including adaptive variants [5] are not well suited to detecting and segmenting text in lecture videos due to varying board types or colours, lighting conditions, handwriting style, writing size and contrast of the writing on the background surface. Recent work favours the u se of machine learning to solve some of these challenges [25]. The Efficient and Accurate Scene Text Detector (EAST) [29] uses a Deep Neural Network (DNN) to detect text. The authors claim it is capable of robustly detecting various forms of text in challenging scenarios such as non-uniform lighting, low-resolution footage and varying orientations. The original EAST model was pre-trained using the International Conference on Document Analysis and Recognition (ICDAR) 2013 and 2015 [2] training datasets.

Automatic Lecture Recording and Virtual Cinematography: Remote learning is a term used to describe the practice of reviewing lecture recordings after the lecture as a supplement to the lecture itself. The automation of this process is referred to as *Automatic Lecture Recording* (ALR) [14], *Virtual Videography* [8] or *Web-based Learning Technology* (WLT) [22] when the automation takes place over the internet. A key feature in ALR systems is the ability to manage what is shown by the camera in a given context [8]. The way the lecture is portrayed is very different from the way a film or television series is recorded and edited. Guidelines ("heuristics"), on the best practices in the lecture recording industry help ALR systems in this regard [10, 15]. **Real-time ALR systems** do all the required processing as the lecture is being recorded [27, 3]. To achieve fast turnaround time for camera changes, a real-time solution often

[2] https://iapr.org/archives/icdar2013/, https://iapr.org/archives/icdar2015/

choose less accurate solutions. Typical implementations of real-time ALR systems have (one or more) PTZ cameras installed in a venue and the decisions made are used to control them. **Post-processing ALR systems** delay decision making until after the video is recorded and then modify the footage to form a new video [9, 11]. Since the video is complete by the time the ALR system runs, the program has access to all frames in the video and can thus make more informed decisions than a real-time solution.

The Virtual Cinematographer is a part of the ALR system and must ensure the final footage displays the most relevant information in an appealing way [10, 11]. The VC includes the Virtual Director (VD) and the Virtual Camera Operator (VCO) to make its deductions about the environment and modify the camera (or camera footage) accordingly. The VD performs scene analysis and decides which areas of the camera footage are most relevant [20, 15]. It sends instructions to the VCO or modifies the footage directly. The VCO manages the recording of the video footage [13, 4].

3 Video Analysis and Virtual Cinematography Framework

This section introduces the design goals of the system followed by the implementation details of the individual modules.

3.1 Design Goals

The design requirements are informed by a current ALR implementation at a university. The system design aims to maximise processing speed on middle-tier equipment without using a GPU. The object detection stage consumes the bulk of the resources and we reduce the CPU and memory footprint by using cheap image differencing-based techniques as our object detection followed by heuristics to mitigate the shortcomings of this differencing approach. Furthermore, the solution aims to be fit for purpose (i.e. should produce output which is sufficiently good for use in an ALR system) and capable of processing any video resolution with support for commonly used video file formats.

To reduce set-up complexities for easy implementation without requiring per-venue setup, parameters should be self-calibrating as much as possible and additional parameters that are configurable or non-trivial to self-calibrate should be stored in a central configuration file. The code of this project will also be released with an open source license to enable further development and usage.

Figure 1 shows the high-level flow of data through our system modules. To facilitate this flow of data and reduce memory requirements, a pipe and filter architecture was identified as the most advantageous for our design goals. The unidirectional flow of this architecture is necessary since data produced by earlier modules is required by later modules in the pipeline. The videos are processed in chunks by front-end modules that reduce the need for large amounts of memory and allow any length video to be processed. These modules were not multi-threaded, however, many of the OpenCV algorithms are already optimised to utilise multi-threading and more than one instance of our system can be instantiated to process multiple videos in parallel, depending on the number of available CPU cores. Lastly, modules can easily be modified, added or removed using this architecture, for any future improvements.

A brief overview of the modules and containers shown in Figure 1 are discussed below. The other core modules are discussed further in their own dedicated sections.

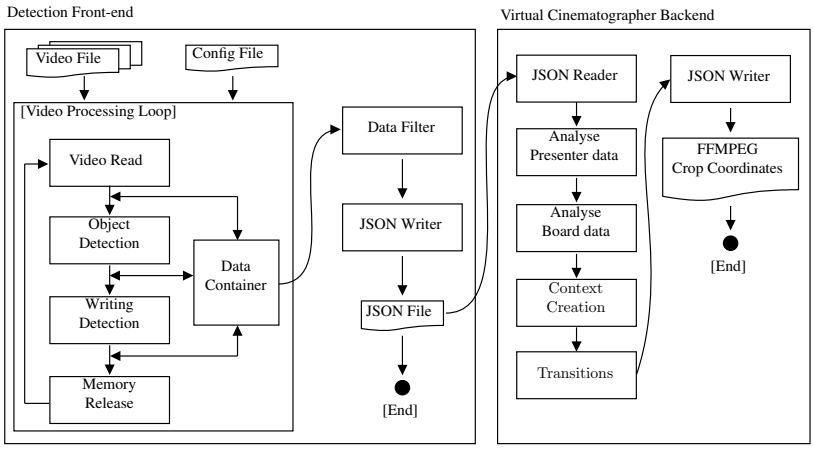


Figure 1: Overview of the complete system architecture.

Configuration file: An external JSON file with customizable default parameter values.
Video processing loop: In this work, a typical 50min 4K, 3840x2160 (width by height in pixels) resolution lecture video recorded at 30fps, a maximum bit rate of 4Mbps and using the H264 codec has an average file size of 2.24GB. Decompressing all frames for analysis would inflate this to an unacceptable 2.7TB of memory. Instead, a small bounded subset of uncompressed frames (a segment) is retained in memory and sequentially processed. Memory is released at the end of the loop and this process repeats until all frames in the input video file have been processed.
Data container: This container (*DataBundle* in our implementation) is passed through all modules providing input data and configuration parameters and stores *MetaFrame* objects to hold the output data produced by each module. The *DataBundle* also contains helper functions to interact with modules.
Meta frame: The *MetaFrame* is a sub-object of the *DataBundle* that stores the object and writing detection output. It contains many helper functions that interface with other modules, exports their data and manages their memory usage.
Video file reader: This module reads, decodes, and downscales the frames from the input video file into memory. The *workingDimension* parameter specifies the target downscaling resolution and *skipFrames* allow frames to be ignored to increase processing speed.
JSON file writer: All presenter, gesture and writing data from the *MetaFrame* objects stored in the *DataBundle* are serialised into the output JSON file.
FFMPEG crop coordinates: The VC outputs a text file of cropping coordinates that are then passed to an adapted version of the original *cropvid* [3] program that uses the *ffmpeg* [4] filters for cropping and rescaling the video frames.
Core modules: The `object detection`, `writing detection`, `data filter` and `virtual cinematographer` modules are discussed in detail in the dedicated sections that follow.

[3] https://github.com/LectureTracking/trackhd/blob/master/cropvid/cropvid.c
[4] https://ffmpeg.org/ffmpeg-filters.html

3.2 Presenter Detection

Object detection is achieved using temporal differencing between two frames, i and $i + (skipFrames + 1)$ where *skipFrames* specifies the number of frames to ignore as an optimization and a means of reducing foreground aperture. A second parameter, *workingDimension*, reduces the input frame resolution to reduce the processing time further. *Otsu thresholding* [17] is used to map this to a binary (foreground/background) image since it has low computational cost. The noise in the binary image is suppressed using three iterations of a morphological opening operation with a 3x3 rectangle-shaped structuring element. The presenter's silhouette is not solid and contains holes due to colour similarities or low motion between the two frames that result in a zero difference. An efficient (8-)connected components analysis (CCA) is used to isolate the blobs of pixels in the binary image corresponding to candidate objects. We then create a bounding box for each blob. The method for storing the rectangles in the *MetaFrame* recursively merges any overlapping rectangles (see Figure 2) using the same merging used for the writing as shown in Figure 8, Appendix A.

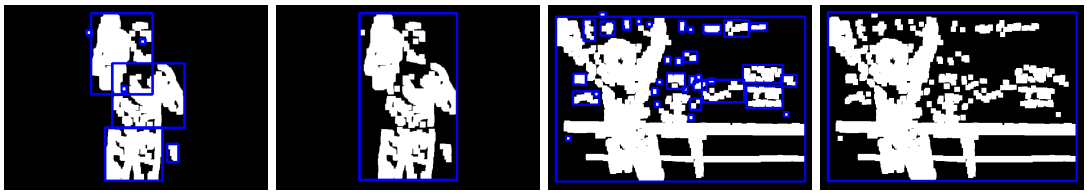
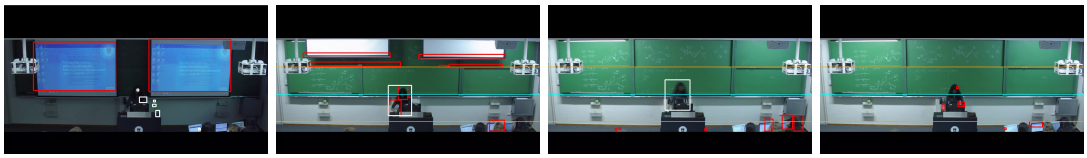


Figure 2: Overlapping bounding boxes are merged recursively.

Data filtering heuristics use information from the three core modules and remove outlying motion that has a low probability of being the presenter, repair missing data and refine the overall output using *adaptive* thresholds.



(a) (b) (c) (d)

Figure 3: Examples scenarios targeted by the data filtering heuristics.

The ***Aspect and area ratio*** is used to flag objects that have abnormal area ratios ($\frac{\text{object area}}{\text{frame area}}$) as shown in Figure 3a and aspect ratios as seen on the projector screens in Figure 3b. *Clustering* the non-deleted current and previous *MetaFrame* object bounding boxes reduces fragmented objects and improves the effects of low presenter motion (see Figure 3b). However, using too many previous frames resulted in oversized clustered boxes. ***Vertical limits*** are calculated using the vertical position (y-coordinate) of the remaining (unflagged) objects (see yellow lines in Figure 3c). Any unflagged objects whose centroid is outside this region is flagged. These objects are typically produced by small amounts of motion such as student heads in view. ***Interpolation*** of the last known (undeleted) object (A) is used to bridge the sequence of frames where there is no valid object (commonly due to low presenter motion as shown in Figure 3). An interpolated box is substituted between (A) and the frame where a valid

object (B) reappears in future. This is also only done if boxes (A) and (B) overlap (potentially indicating a stationary presenter).

3.3 Writing Detection

Text detection is achieved using the EAST text detector with the pre-trained model proposed by Zhou et al. [29]. Since we do not use the GPU accelerated version, run-times are no longer real-time, but we need only run the detection infrequently since writing/text changes slowly over time. The *writingDetectionSkipTime* parameter specifies the frequency (in seconds) of detections. The output produced by the EAST network are rotated rectangles which typically bound individual words. However, board writing generally consists of collections of words. We thus use a *clustering algorithm* to group words that are close to each other. These cluster regions and the number of rotated rectangles in each cluster are output by this module, indicating regions of potential text. DBSCAN clustering [6] was chosen to cluster the words since it does not require the number of clusters to be specified upfront. Good clustering results are obtained by using the four vertices of the boxes. Clustering based on word box vertices often results in word boxes belonging to two separate clusters or not fully enclosed by the cluster boundary. These clusters are merged using the approach shown in Figure 8, Appendix A.

3.4 Virtual Cinematographer

The VC (see Figure 1) must read in the information from the front-end of the pipeline (JSON format) and build its own internal memory representation. This reduces the run time of the VC and allows it to manipulate the data later without modifying the source. The VC must identify and locate the presenter in each 4K frame. This is not always possible, however, since the presenter can walk out of the camera's field of view at the time of recording or the front of the venue may become too crowded for the front-end to identify the presenter. In these cases the frame will contain no presenter data. Before anything else can be done, the VC must fill in these "gaps" in the recording. The VC does this by (first) writing the last known position of the presenter into the frame containing no presenter data.

The VC uses information about the boards in the lecture venue to identify what is most important in the scene. This step in the pipeline is where the VC analyses the tracking information about the boards to see which boards were used in the venue and at what times in the video. The VC identifies which boards were used by tracking the number of features on each board over time: a large change in the number of features on a board over a small number of frames suggests that a board is in use. It may also be that the presenter has moved in front of the board, but the board would still be important since the presenter could refer to (or modify) the board content during this time. If the presenter is moving past the board to get to another area of the venue, then it is still useful to include the board as part of the context for as long as the presenter is in front of it.

Once the VC has identified the presenter in every frame and highlighted active boards and the usage frame intervals, the VC can separate the video into sections we called *contexts*. These contexts are rectangles that represent the position and size of the cropping window for a section of the video. They represent a change in the framing choice made by the VC. The VC includes the presenter and any active boards as part of the context for each frame in the video. An example of a context can be seen in the third image of Figure 2 where the presenter is standing in front of the board as it is being used. The Presenter has occluded some of the features on the board and is also actively modifying them. This requires both the presenter and the board to be highlighted as important venue elements on this frame in the video.

Once the VC has created the contexts for each frame, it must find a way to move the cropping window from one context to the next such that the resultant video does not appear jarring to viewers. It does this by creating *transitions* which move or resize the cropping window gradually so viewers perceive it as camera motion when, in fact, the video was recorded using a stationary camera. Each transition has 4 movements to make:
1. The horizontal motion for the top left corner of the cropping window.
2. The vertical motion for the top left corner of the cropping window.
3. The horizontal motion for the bottom right corner of the cropping window.
4. The vertical motion for the bottom right corner of the cropping window.

With these 4 movements the VC is able to pan horizontally, vertically, and zoom in or out with a single movement of the cropping window to minimise the conspicuousness of the transition to the viewer. Each of these movements take place on every transition where the VC interpolates the cropping window across all the frames of the transition (including the frames that were skipped by the front-end). This interpolation is done using the function $\cos(x + \pi) + 1$ using the domain $[0, \pi]$ radians. This curve allows a gradual acceleration and deceleration of the cropping window across the frames in the transition. Once the VC has finalised all transitions, this information is recorded in the output file (JSON format) for a third-party cropping process. These crop coordinates are the final output of the VC.

3.5 Implementation Language and Libraries

For efficiency, the project is implemented in C++. We used the Open Computer Vision (OpenCV) library [5] for image processing. Additional algorithms were drawn from the Mlpack [6] open source library and statistical functions from the Boost [7] library.

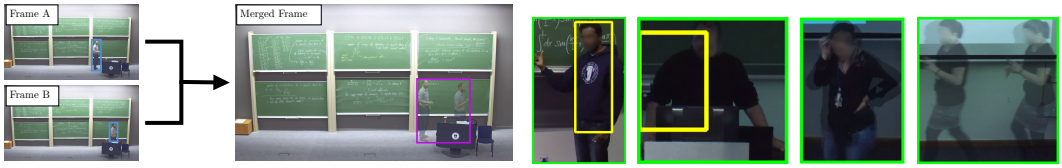
4 Experimental Setup

Figure 4: Images showing a sample of different venue layouts.

Data Source: The dataset consists of 25 lecture videos (each about $45 - 50$ minutes long) from different venues. An example of the diversity of lecture venue layouts can be seen in Figure 4.
Evaluation Hardware: *Intel® Core™ i5-4670K CPU @ 3.40GHz x 4* with a *Western Digital WDC WD10EZEX 1TB HDD @ 7200rpm* and *2 x 8GB DDR3 RAM (16GB total).*
Ground truth (GT) data: Using a purposeful sampling approach, we selected 100 (each roughly 2-minute length) video clips from the dataset covering a diverse range of behaviours and environments. These clips were annotated frame-by-frame using the Computer Vision

[5] https://opencv.org/
[6] https://www.mlpack.org/
[7] https://www.boost.org/

Annotation Tool (CVAT) [8]. The ground truth dataset is further partitioned into training ($n = 76$) and testing ($n = 24$) sets such that both sets have a similar distribution of objects and attributes. The training set is used to explore and calibrate system parameters while the test set is reserved for evaluating the accuracy of our system after parameter calibration. Each frame was annotated with good (e.g. Figure 5a) or low (e.g. Figure 5b) lighting. All person objects were annotated and the presenter was labelled. The tightest bounding box enclosing all content on a board/screen was used and the attributes shown in Table 2, Appendix A were assigned to each box where applicable.



(a) (b)

Figure 5: (a) Ground truth *MetaFrame* creation for evaluation. (b) Example of $IOU = 0.3$ between ground truth (green) and system (yellow) detections. Last two images show examples of "low" and "high" presenter motion respectively.

Presenter Detection: All the *MovingEntity* objects in each *MetaFrame* are compared to the equivalent GT *MetaFrame*. Since all the GT videos were labelled per frame, the final ground truth presenter bounding box is the union of bounding boxes (between the two frames that were used to produce the original *MetaFrame* object in our system) as illustrated in Figure 5a.

We use the popular Intersection over Union (IOU) metric [18] to compare the system predictions to the GT for each *MetaFrame*. We have adapted and integrated their evaluation code [9] into our evaluation implementation. A minimum IOU threshold of 0.5 is typically used, however, we do not require precise overlap but rather an approximate location of the object. Using an IOU of 0.3 achieves this goal for evaluation and Figure 5b shows two examples of an $IOU = 0.3$. Furthermore, our implementation does not provide a frame-by-frame prediction but instead over a time-slice since each system prediction is a region of motion across a few frames determined by the *skipFrames* parameter. The GT presenter motion is calculated using the x coordinate of the bounding box centroid between frames A and B in Figure 5a.

Writing detection: We followed the evaluation protocol proposed by Wolf and Jolion [21] for evaluating our writing detection module, since it accounts for fragmented detection (one-to-one, one-to-many and many-to-one). Our method does not aim to achieve 100% correct overlaps and the annotated frames only provide the tightest bounding box per board, and are likely to fail stringent overlap tests.

Virtual cinematographer: The user evaluation layout was designed to display two video clips side-by-side to participants. Both video clips were taken from the same timestamp in the same lecture video, but the VC's configuration settings were altered to produce different camera movements in the final output video.

Participants were prompted to select the video from the pair which they preferred based only on watching the videos. Once a video was chosen, participants were then asked eight questions (see Table 4 of Appendix A). These questions were gathered from previous studies

[8] https://github.com/openvinotoolkit/cvat

[9] https://github.com/rafaelpadilla/review_object_detection_metrics/blob/main/src/evaluators/pascal_voc_evaluator.py

in the literature [15, 20] and were used to determine why participants chose one video and not the other. These questions were phrased as affirmations and participants were provided with a fixed scale of discrete responses ranging from 'Strongly Disagree' to 'Strongly Agree'.

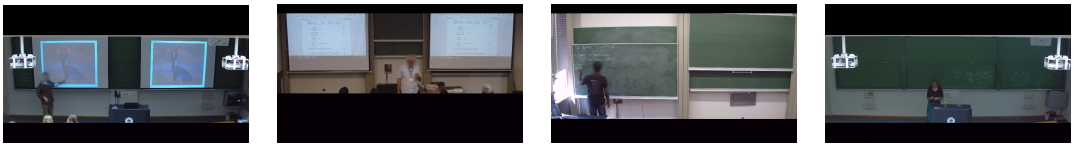


Figure 6: Images showing the different venue layouts used in the user evaluation of the VC

The VC is forced to accommodate a variety of venue layouts. In the user evaluation we focused on the four layouts in figure 6 to judge the VC's output across a range of challenging inputs. These venues show a combination of bright projection surfaces, movable blackboards, variable lighting conditions and 'distractor' motion (such as student heads). The venues are detailed in Table 5 of Appendix A. Note that the video inputs include a privacy mask to protect the identity of the audience and to remove parts of the venue that are not relevant to the lecture. This constrains the VC's framing decisions, as aspect ratio must be maintained.

The session lasted less than an hour (in total) for each participant. The evaluation included 20 video pairs with the follow-up questions after each pair (making a total of 40 questions). The evaluation was hosted online, so participants could take part remotely and could pause the evaluation to return later. We collected 55 complete submissions from participants and extracted the data (CSV) to process our results.

5 Results

This section discusses the results and observations for the the detection and VC modules.

5.1 Presenter Detection

Parameter calibration: The training dataset was used to empirically select default parameter values. We chose *skipFrames* = 28 as the default (see Figure 9a, Appendix A). Figure 9b in Appendix A shows the effect of downscaling the input 4K frame resolution on runtime and *F1-score*. The runtime averages are with respect to the 2-minute video clips in our dataset. The video read time includes the downscaling operation time and there is a gradual increase in this time as we reduce the video size, which is expected. The downscaling operation adds minimal time with a great reduction in runtime of the other processing modules. We selected *workingDimension* = 720*p* as it provided a balance between presenter detection and writing detection outcome as shown in Figure 9c, Appendix A.

Observations: A Bayesian Multivariate Binary Logistic Regression Model with the Bernoulli family using the *brms* package [1] in R [10] was used to evaluate the association between environmental factors and observing an *F1-score* of 1 for each frame. In Appendix A, the odds ratios are shown in Table 1 and the overall *F1-score* under each condition is shown in Table 3a. The odds ratio indicates the likelihood of observing an *F1-score* = 1 (perfect outcome) for a frame under each condition. We observed the filter module improved the results under all scenarios except scenes with multiple objects. The overall results improved from *F1-score* = 0.51 to

[10] https://www.R-project.org/

F1-score = 0.78. Low lighting and distractors (student heads) had minimal to no effect on our detection as most cases were mitigated by the filter heuristics. Frames with multiple objects significantly reduced detection accuracy and were not improved by the filter. These occurrences are typically short in duration due to students entering/leaving the venue and does not have any substantive effect on the VC, since it resorts to a wider zoom in persistently busy scenes. The filter heuristics were also able to improve low motion, however, high presenter motion naturally produced better detection results due to better silhouettes being produced during the differencing operation.

5.2 Writing Detection:

Parameter calibration: Figure 9d, Appendix A shows the effect of the *writingDetectionSkipTime* parameter on the average runtime and *F1-score*. There is a marginal difference in *F1-score* which is expected, since the *writingDetectionSkipTime* parameter only influences the frequency of detection and any variations in *F1-score* are just due to different text at those times. We empirically selected an interval of 30 seconds for writing detection. We also compared the outcome of the writing detection *F1-score* when using the auto calibrated threshold discussed in Section 3.3 as opposed to manually selecting an optimal threshold. The manual calibration only produces a very small improvement from 0.554 to 0.557, indicating that the chosen method works well.
Observations: The odds ratios for the writing detection factors can be seen in Table 1 and overall *F1-score* scores in Table 3b of Appendix A. We observe an very large positive impact of projector text on the outcome, while poor writing contrast and diagrams have a significantly negative impact on outcome (see Figure 7 in Appendix A for example output). EAST only detects text and the improvement with projector text and the degradation with diagrams is expected. Projector text is typically high contrast, unlike handwriting on dusty boards and is expected to perform better. Figure 10b and Figure 10a in Appendix A show the overall performance graphs of the writing detection. A recall of $49 - 57\%$ is achieved ($t_r = 0.8$) and only declines rapidly after about 70% of the precision constraint is reached. As $t_r \to 1$, object recall does not reach 0 indicating that many detected boxes are smaller than the ground truth boxes. This is expected: the GT data only provides a single bounding box enclosing all text on each board.

The writing information is supplementary input to the VC to enhance framing decisions and there is minimal impact on the VC output when writing detection degrades. In such cases the VC resorts to focusing on the presenter.

5.3 Virtual Cinematographer

We tested 2 aspects of the video which affect the final video created from the VC's cropping data. The VC's configuration settings affected its decision making and the venue layout affected the VC's cropping data and the resultant video indirectly.
Tables and figures: The results are visualised in the graphs of Figure 11 for changing VC configurations and Figure 12 for changing venue types (in Appendix A). The questions participants were asked after each video choice is included in Table 4 in Appendix A.
Observations: We notice that, for the first 3 questions regardless of configuration type or the venue layout, the majority of participants (more than 75%) responded either "Stongly Agree", "Agree", or "Neutral". This means that the VC tracked the presenter smoothly, showed participants what they wanted to watch, and changed shots at an acceptable frequency for most participants (according to the questions in Table 4). We also notice that, for questions 4 - 8, the

majority (more than 62.5%) of participants responded either "Strongly Disagree", "Disagree", or "Neutral" regardless of VC configuration or venue type. Question 4 is phrased generally (as it was phrased in the original papers [20, 15]), and was used to gauge the general impressions of participants regarding the VC's camerawork. The remaining questions were based on specific venue layouts which affected the results accordingly. We see for Question 5 that there is a better reception in Venue 3 than in any other venue. This is because the presenter was writing in a large print on the boards and participants could follow along more easily.
We tested the runtime of the VC from the time it reads in the tracking information until it produces cropping information as output and we found that it takes an average of 40.3 ms to run a 2-minute test video (and, given the linear nature of the process, we can expect a runtime of approximately 1.4 seconds for a 45-minute lecture video). We also tested the reduction in filesize from the original 4K video to the cropped 720p video and found that the mean reduction was 81.3%, which means the output video is only 18.7% of the input video (on average).

Figure 13 in Appendix A shows a box plot of runtimes for each of the modules (produced from a smaller sample of videos, $n = 88$, to save time). Since the VC only generates and exports cropping rectangles and does not read in video frames, it has an extremely fast runtime. The detection and cropping modules process video frames and a large portion of time is used for disk read/write operations. Using an solid state disk could reduce these runtimes. From the data in the plot, we can conclude that the worst case total runtime is less than 145 seconds to fully process a 2-minute video via the full pipeline (or an additional 50% overhead on top of the input video length).
Caveat on participants: We also notice from Figure 14 in Appendix A that many participants had little experience with lecture videos, and thus were not sure what to choose. We can therefore attribute some of the negative reception to the lack of experience with watching lecture videos in our participant pool, since many of these participants would not know what to expect from lecture recordings and, therefore, gave the VC a bad score based on their exposure to other kinds of videos. Further work is required to elaborate on this aspect of the user evaluation.

6 Conclusions

This work presents a system that post-processes 4K video streams to a reduced size, suitable for lower bandwidth environments using only middle tier hardware with no GPU. The front-end extracts presenter motion and writing surfaces and feeds these to the back-end VC to allow sensible framing decisions.

Presenter detection was shown to be sufficient for our application (*F1-score* = 0.78) even under low lighting conditions. While multiple moving objects (student's moving through venue) reduce presenter tracking accuracy, these events are typically transient and the VC is able to gracefully zoom out to a wider field of view. Although student heads were constant distractors in the predecessor system, the new system is largely unaffected. This success is attributed to the data filter heuristics for ignoring the distractors. **Writing detection** achieved satisfactory results: while poor contrast and diagrams were a major factor reducing accuracy, the VC was not negatively affected and defaulted to a presenter-only focus. Poor detection of diagrams is a limitation of EAST and scenes with text only produced significantly better results.

The Virtual Cinematographer took tracking information from the front-end and produced cropping information that was used to produce the output video. Our user evaluation of the VC determined that the questions that received a largely negative reception were specific to the varying factors in a venue. The overall impressions, however, were that it was fit for

purpose. Videos produced by the VC show viewers what is most important, while changing shots at a satisfactory frequency, and moved the camera without jarring the view. We also found that the majority of participants in the user evaluation had little experience with lecture videos, which could be a contributing factor to the VC's poor reception. We expect the VC to operate in ≈ 1.4 seconds for a 45-minute input lecture video (as mentioned in our observations above) and the file size of the output video is greatly reduced from that of the input file size.

The system can be improved in several ways. Additional heuristics could be developed to improve the robustness of segmentation of the temporal differencing output. Pointing gesture detection could also be explored, to provide additional context to the VC. Further VC user studies with more experienced lecture audiences could be conducted to obtain more conclusive results.

7 Acknowledgments

We would like to thank the Centre for Innovation in Learning and Teaching (CILT) at the University of Cape Town for providing access to the lecture recordings used in this work. We would also like to thank the lecturers who agreed to allow their lecture recordings to form part of this study. Additionally, we would like to thank the data annotators for meticulously annotating these lecture recordings as ground truths for evaluating the front-end module. Finally, we thank the participants in the user evaluation of the VC for volunteering their time and effort to the study.

References

[1] P. C. Bürkner. "brms: An R Package for Bayesian Multilevel Models Using Stan". In: *Journal of Statistical Software* 80.1 (Aug. 2017), pp. 1–28.

[2] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. "OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.1 (July 2019), pp. 172–186.

[3] A. Cavallaro, R. Chandrasekera, and M. Taj. "Hands-On Experience in Image Processing: The Automated Lecture Cameraman". In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*. Vol. 3. ICASSP '07. Honolulu, HI, USA: IEEE, 2007, pp. 721–724.

[4] D. Cymbalák, O. Kainz, and J. František. "Real-Time Automatic Selection of the Best Shot on Object in 4K Video Stream Based on Tracking Methods in Virtual Cropped Views". In: *International Journal of Computer and Electrical Engineering* 7.4 (2015), pp. 275–282.

[5] K. Davila and R. Zanibbi. "Whiteboard Video Summarization via Spatio-Temporal Conflict Minimization". In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Kyoto, Japan: IEEE, July 2017, pp. 355–362.

[6] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*. AAAI, 1996, pp. 226–231.

[7] S. French and G. Kennedy. "Reassessing the value of university lectures". In: *Teaching in Higher Education* 22.6 (Aug. 2017), pp. 639–654.

[8] M. Gleicher and N. Ferrier. "Evaluating Video-Based Motion Capture". In: *Proceedings of Computer Animation 2002 (CA 2002)*. Geneva, Switzerland: IEEE, 2002, pp. 70–80.

[9] M. Gleicher and J. Masanz. "Towards Virtual Videography (Poster Session)". In: *Proceedings of the eighth ACM international conference on Multimedia*. MULTIMEDIA '00. Marina del Rey, California, USA: ACM, 2000, pp. 375–378.

[10] M. L. Gleicher, R. M. Heck, and M. N. Wallick. "A Framework for Virtual Videography". In: *Proceedings of the 2nd International Symposium on Smart Graphics*. Vol. 22. SMARTGRAPH '02. Hawthorne, New York, USA: ACM, 2002, pp. 9–16.

[11] R. Heck, M. Wallick, and M. Gleicher. "Virtual Videography". In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 3.1 (2007), 4–es.

[12] W. J. Heng and Q. Tan. "Content enhancement for e-learning lecture video using foreground/background separation". In: *2002 IEEE Workshop on Multimedia Signal Processing*. Institute of Electrical and Electronics Engineers Inc., 2002, pp. 436–439.

[13] F. Lampi, S. Kopf, M. Benz, and W. Effelsberg. "An Automatic Cameraman in a Lecture Recording System". In: *Proceedings of the International Workshop on Educational Multimedia and Multimedia Education*. Emme '07. Augsburg, Bavaria, Germany: ACM, 2007, pp. 11–18.

[14] H.-C. Liao, M.-H. Pan, M.-C. Chang, and K.-W. Lin. "An Automatic Lecture Recording System Using Pan-Tilt-Zoom Camera to Track Lecturer and Handwritten Data". In: *International Journal of Applied Science and Engineering* 13.1 (2015), pp. 1–18.

[15] Q. Liu, Y. Rui, A. Gupta, and J. J. Cadiz. "Automating Camera Management for Lecture Room Environments". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '01. Seattle, Washington, USA: ACM, 2001, pp. 442–449.

[16] A. Mavlankar, P. Agrawal, D. Pang, S. Halawa, N. M. Cheung, and B. Girod. "An interactive region-of-interest video streaming system for online lecture viewing". In: *2010 18th International Packet Video Workshop*. Hong Kong, China: IEEE, 2010, pp. 64–71.

[17] N. Otsu. "A Threshold Selection Method from Gray-Level Histograms". In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (1979), pp. 62–66.

[18] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto, and E. A. B. d. Silva. "A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit". In: *Electronics* 10.3 (Jan. 2021).

[19] K. Risha and A. C. Kumar. "Novel Method of Detecting Moving Object in Video". In: *Procedia Technology* 24 (Jan. 2016), pp. 1055–1060.

[20] Y. Rui, L. He, A. Gupta, and Q. Liu. "Building an intelligent camera management system". In: *Proceedings of the ACM International Multimedia Conference and Exhibition*. MULTIMEDIA '01. Ottawa, Canada: ACM, 2001, pp. 2–11.

[21] C. Wolf and J. M. Jolion. "Object count/area graphs for the evaluation of object detection and segmentation algorithms". In: *International Journal of Document Analysis and Recognition (IJDAR)* 8.4 (Sept. 2006), pp. 280–296.

[22] K. Woo, M. Gosper, M. McNeill, G. Preston, D. Green, and R. Phillips. "Web-based lecture technologies: blurring the boundaries between face-to-face and distance learning". In: *ALT-J* 16.2 (June 2008), pp. 81–93.

[23] X. Wu, D. Sahoo, and S. C. Hoi. "Recent advances in deep learning for object detection". In: *Neurocomputing* 396 (July 2020), pp. 39–64.

[24] B. Wulff and A. Fecke. “LectureSight - An open source system for automatic camera control in lecture recordings”. In: *2012 IEEE International Symposium on Multimedia*. Irvine, CA, USA: IEEE, 2012, pp. 461–466.

[25] Q. Ye and D. Doermann. “Text Detection and Recognition in Imagery: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.7 (July 2015), pp. 1480–1500.

[26] T. Yokoi and H. Fujiyoshi. “Virtual camerawork for generating lecture video from high resolution images”. In: *2005 IEEE International Conference on Multimedia and Expo*. Amsterdam, Netherlands: IEEE, 2005.

[27] C. Zhang, Y. Rui, J. Crawford, and L.-W. He. “An Automated End-to-End Lecture Capture and Broadcasting System”. In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 4.1 (2008), pp. 1–23.

[28] Q. Zhang, H. Sun, X. Wu, and H. Zhong. “Edge Video Analytics for Public Safety: A Review”. In: *Proceedings of the IEEE* 107.8 (July 2019), pp. 1675–1696.

[29] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. “EAST: An Efficient and Accurate Scene Text Detector”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI, USA: IEEE, July 2017, pp. 2642–2651.

A Appendix

Presenter detection				
	Est.	Est.Error	Q2.5	Q97.5
Intercept	5.98	1.20	4.17	8.52
LightingLow	0.42	1.29	0.26	0.69
PMLevelHigh	2.18	1.06	1.94	2.46
MultipleObjectsYes	0.04	1.54	0.01	0.08
StudentHeadsYes	1.05	1.20	0.74	1.50

Writing detection				
	Est.	Est.Error	Q2.5	Q97.5
Intercept	0.21	1.98	0.05	0.69
ProjectorTextYes	32.88	2.97	4.69	341.18
PoorContrastYes	0.01	6.50	0.00	0.14
DiagramsYes	0.07	2.49	0.01	0.40

Table 1: Exponentiated parameter estimates with the estimated error, lower and upper 95% credibility interval for the presenter (Left) and writing (right) detection environmental factors.

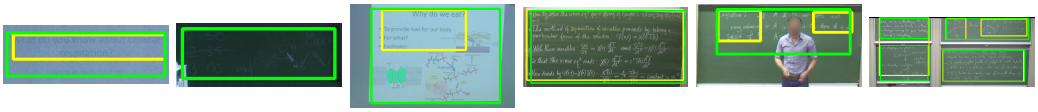


Figure 7: Ground truth is shown with the green rectangle and system with yellow. Poor visibility and diagrams affects the detection (first three images). Good detections and fragmentation can be seen in the last three images.

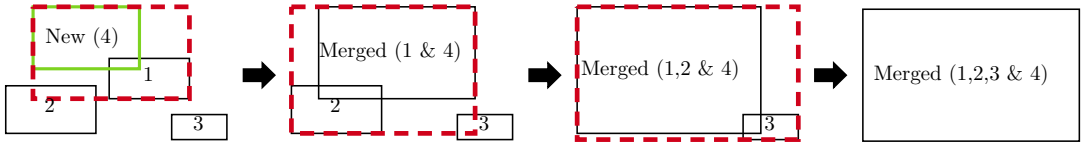


Figure 8: When a newly added cluster rectangle (green) intersects with an existing cluster (1) they are merged as shown by the dashed red rectangle. This newly created merge intersects with existing rectangle (2) and the process is repeated until all intersections are merged.

Attribute	Values	Description
Writing type	Text/Handwriting	Projector text or handwriting on a board.
Poor contrast	Yes/No	Low text visibility.
Shadow	Yes/No	If a shadow is cast onto any of the enclosed text.
Characters	Yes/No	Standalone letters, symbols or equations/expressions.
Diagrams	Yes/No	Contains diagrams, drawings or images.

Table 2: Attributes used for annotating writing in the ground truth data.

Attribute	Value	F1
Lighting	Low	0.70
	Good	0.82
Student Heads	No	0.79
	Yes	0.74
Multiple Objects	No	0.80
	Yes	0.43
Presenter Motion	Low	0.68
	High	0.88

(a) Presenter detection

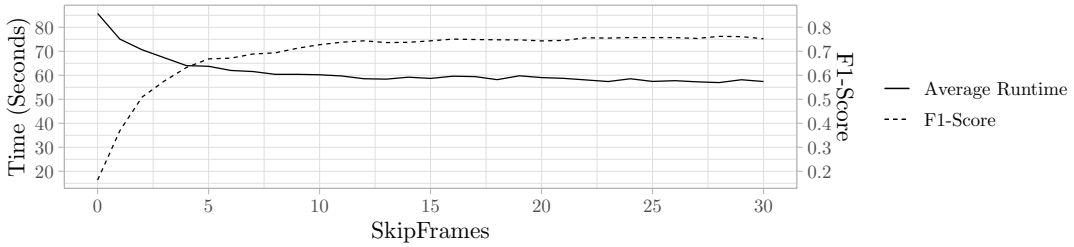
Attribute	Value	F1
Poor Contrast	No	0.62
	Yes	0.19
Diagrams	No	0.62
	Yes	0.39
Projector Text	No	0.51
	Yes	0.57

(b) Writing detection

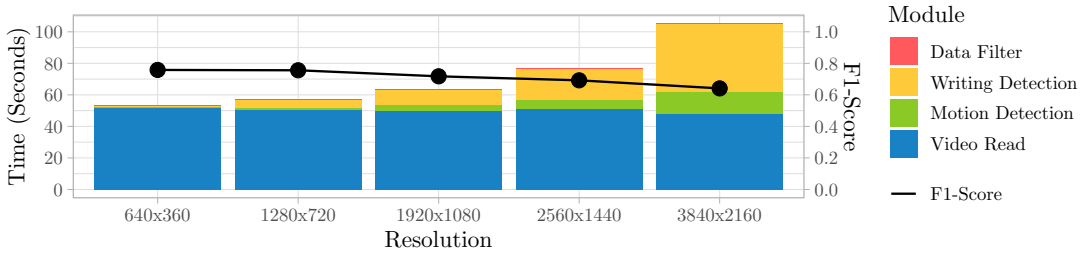
Table 3: Overall *F1-score* for the presenter and writing detection under various conditions reported individually.

1	The camera operator tracked the presenter smoothly
2	I could see what I wanted to watch
3	I like the frequency with which the camera shots changed
4	Overall, I liked the way the operator controlled the camera
5	I was able to follow with what was written on the board
6	The camera view was zoomed and centred appropriately
7	I was able to see the presenter's facial expressions
8	I was able to see the presenter's gestures

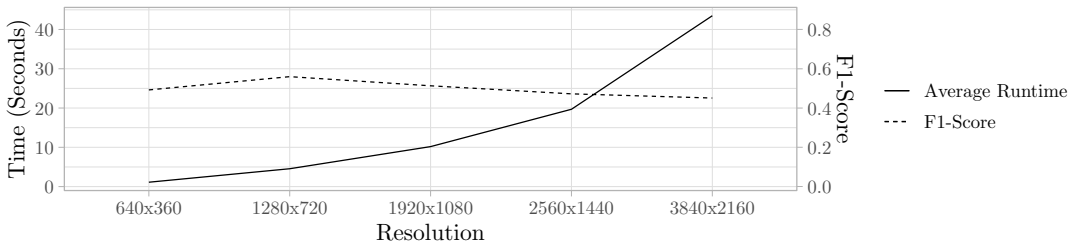
Table 4: Questions used in the VC User Evaluation after each video pair



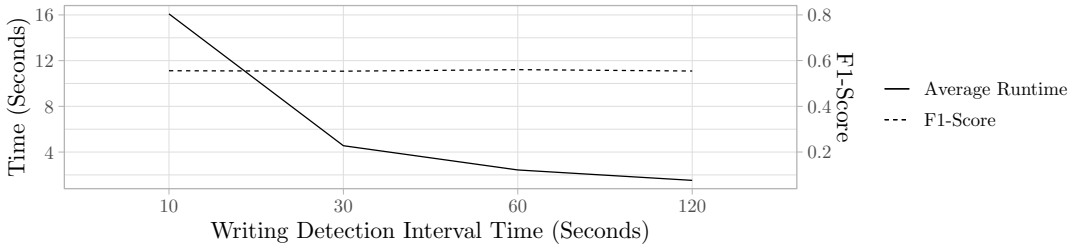
(a)



(b)



(c)



(d)

Figure 9: Effects of (a) *skipFrames* vs. overall front-end runtime and *F1-score*. (b) *workingDimension* vs. module runtime and *F1-score*. (c) *workingDimension* parameter vs. writing detection runtime and *F1-score*. (d) *writingDetectionSkipTime* vs. writing detection runtime and *F1-score*.

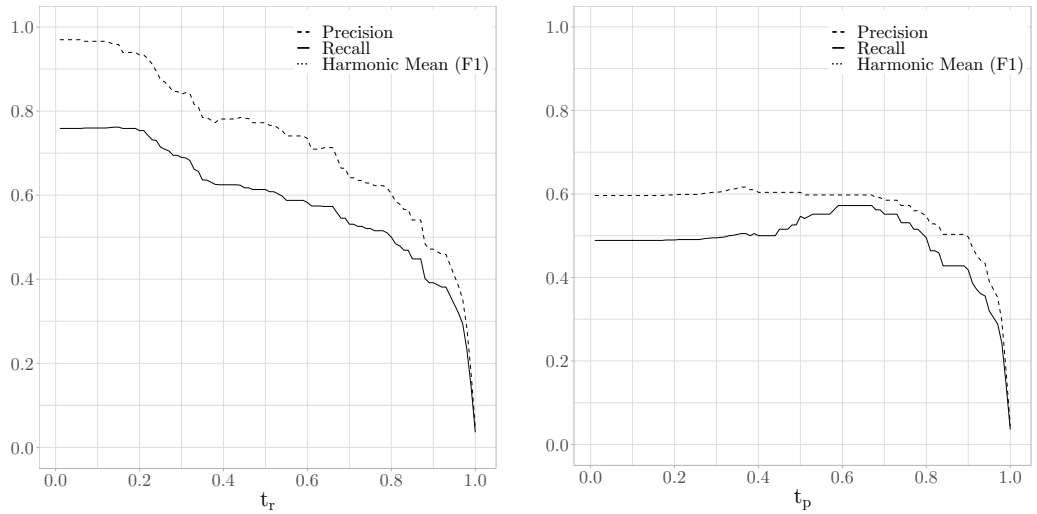


Figure 10: Overall performance graph of the writing detection module. Figure (a) shows varying area recall constraint t_r while fixing $t_p = 0.4$ and (b) varying area precision constraint t_p while fixing $t_r = 0.8$.

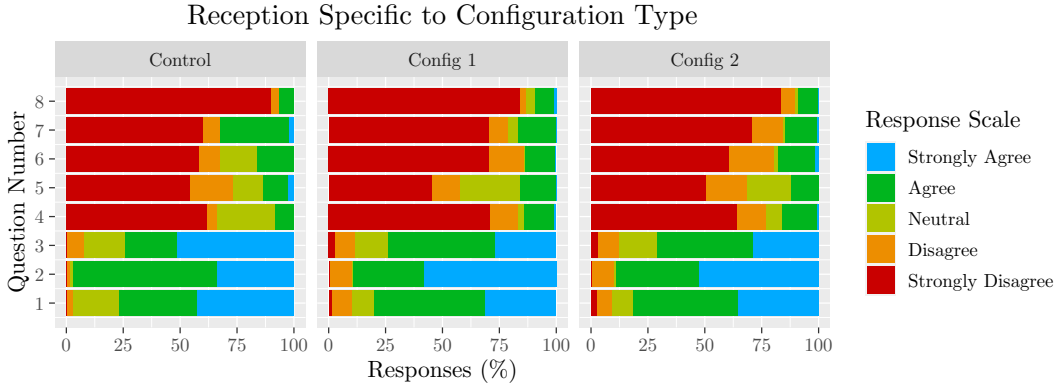


Figure 11: Responses from participants during the evaluation of the VC pertaining to the VC configuration type

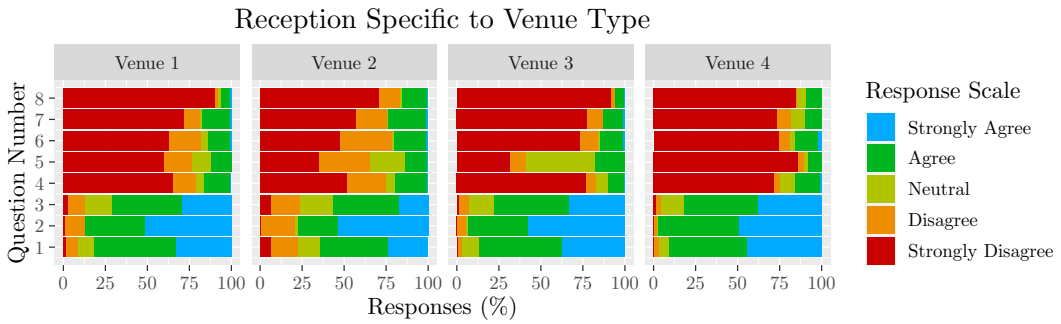


Figure 12: Responses from participants during the evaluation of the VC pertaining to the VC venue type

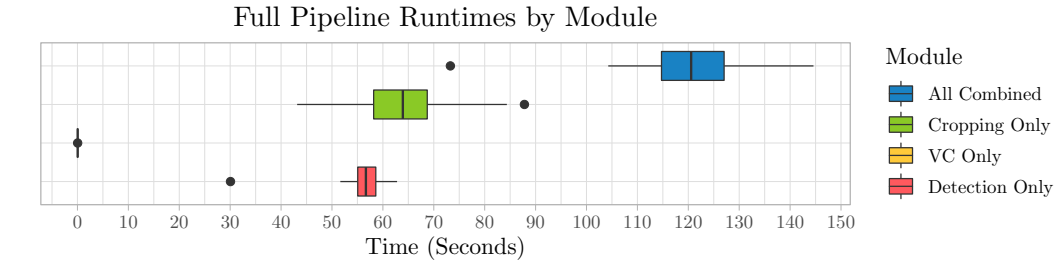


Figure 13: Distribution of runtime per module and the combination of them all on a sample set ($n = 88$) of 2-minute video clips.

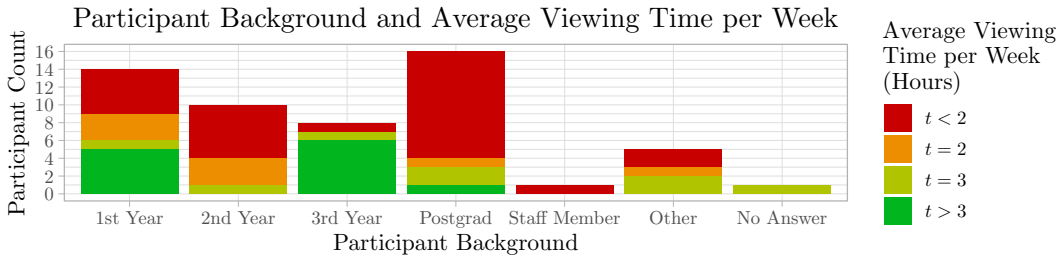


Figure 14: Participant background and average viewing time per week

	See Figure 6. Left to right Venues 1, 2, 3, and 4
Venue 1	There are 2 projector screens, a wide space for pacing, 2 large projectors which affect the camera's lighting, and a few students in the front row. Privacy masks protect student identities and remove irrelevant regions of the venue. The VC can no longer zoom out fully because the height is reduced by the privacy masks and the aspect ratio must be preserved.
Venue 2	This layout is similar to Venue 1, but the projector screen content is small and hard to read. The audience is also more active, which makes it difficult to identify the presenter reliably.
Venue 3	This layout has a narrower space for the presenter to pace. The boards are large and close to the camera, which makes the writing easier to read. No students are visible in this venue. The boards can move vertically, which gives the front-end difficulty (both) with board usage detection and in maintaining the continuity of the boards when they are moved.
Venue 4	This layout has 2 large, bright projectors in view and a thick privacy mask at the top of the venue. The presenter also does not pace in this video despite having space for it. By staying near the podium, the presenter's bounding box is combined with that of the podium every time they overlap, forcing the VC to remain zoomed out far enough to include both. All these layouts were chosen because they posed a unique set of challenges for the VC which we could use to test how the VC copes with varying venue layouts.

Table 5: Here we describe the Venue Layouts shown in the user evaluation