Fast Presenter Tracking for 4K Lecture Videos using Computationally Inexpensive Algorithms

---

By:
Charles Fitzhenry

Supervised By:
A/Prof. Patrick Marais & Stephen Marquard

Dissertation presented for the degree of

Master of Science

in the

Department of Computer Science

University of Cape Town

July 2022

Declaration

I know the meaning of plagiarism and declare that all of the work in the dissertation, save for that which is properly acknowledged, is my own.

Abstract

Lecture recording has become an essential tool for educational institutions to enhance the student learning experience and offer online courses for remote learning programs. High-resolution 4K cameras have gained popularity in these systems due to their affordability and clarity of written content on boards/screens. Unfortunately, at 4K resolution, a typical 45-minute lecture video easily exceeds 2GB. Many video files of this size place a financial burden on institutions and students, especially in developing countries where financial resources are limited. Institutions require costly high-end equipment to capture, store and distribute this ever-increasing collection of videos. Students require a fast internet connection with a large data quota for off-campus viewing, which can be too expensive for many, especially if they use mobile data.

This project designs and implements a low-cost presenter and writing detection front-end that can integrate with an external Virtual Cinematographer (VC). Gesture detection was also explored; however, the frame differencing approach used for presenter detection was not sufficiently robust for gesture detection. Our front-end is carefully designed to run on commodity computers without requiring expensive Graphics Processing Units (GPU) or servers. An external VC can use our contextual information to segment a smaller cropping window from the 4K frame, only containing the presenter and relevant boards, drastically reducing the file size of the resultant videos while preserving writing clarity. The software developed as part of this project will be available as open source.

Our results show that the front-end module is fit for purpose and sufficiently robust across several challenging lecture venue types. On average, a 2-minute video clip is processed by the front-end in under 60 seconds (or approximately half of the input video duration). The majority (89%) of this time is used for reading and decoding frames from storage. Additionally, our low-cost presenter detection achieves an overall F1-Score of 0.76, while our writing detection achieves an overall F1-Score of 0.55. We also demonstrate a mean reduction of 81.3% in file size from the original 4K video to a cropped 720p video when using our front-end in a full pipeline with an external VC.

Acknowledgements

Thank you to my Lord and Saviour, Jesus Christ, for His endless love, grace and guidance and for providing me with the means and strength to complete this project.

I am truly blessed to have such wonderful parents who have always stood steadfast by my side and supported me in everything I do. Thank you for your endless love and support, both financially and emotionally, that has enabled me to pursue this master's degree. Thank you for always believing in me and encouraging me to persevere during this project until completion.

To my brother, thank you for all your love and immense support, continuous encouragement, and everything you've done to help me complete this project. I'm incredibly blessed to have you as my brother.

To my mother and brother, thank you for sacrificing countless hours of your time for the monotonous task of annotating my ground truth videos, which were essential for my project. Your meticulous attention to detail is unrivalled, and no "Mechanical Turk" would have had the interest of my project at heart as you did!

Thank you to my friend Tanweer Khatieb for your work on the Virtual Cinematographer, which was used in testing the output of this project, as well as all your support, contributions and many whiteboard sessions that were so remarkable throughout this project.

I am very blessed and privileged to have such excellent supervisors, A/Prof. Patrick Marais and Stephen Marquard. This project would be impossible without your support, guidance and encouragement. Thank you for all your kindness, patience and the enormous amount of time devoted to this project, along with all the valuable feedback you have provided me. I would also like to thank A/Prof. Patrick Marais for funding the equipment, statistical consulting, conference fees and all other costs associated with this research. Also, thank you to Stephen Marquard and the Centre for Innovation in Learning and Teaching for providing access to their lecture recording systems and the videos used in this project, which were vital for my project. Thank you to the lecturers who agreed to allow their lecture recordings to form part of this research.

Thank you to my friends, colleagues in the lab, staff in the Department of Computer Science, the Faculty of Science, the University of Cape Town and everyone else who contributed to this journey. The impact of this experience will forever be imprinted on my life and career.

Contents

Figures

Tables

Listings

Acronyms

4K	3840 x 2160 resolution
ALR	Automated Lecture Recording
BRMS	Bayesian Regression Models using Stan
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRAFT	Character Recognition Awareness For Text detection
CVAT	Computer Vision Annotation Tool
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DL	Deep Learning
DNN	Deep Neural Network
EAST	Efficient and Accurate Scene Text detector
FN	False Negative
FP	False Positive
FPS	Frames Per Second
GPU	Graphics Processing Unit
GT	Ground Truth
H0	Null Hypothesis
HDI	Highest Density Intervals
HMM	Hidden Markov Model
ICDAR	International Conference on Document Analysis and Recognition
IO	Input Output
IOU	Intersection Over Union
JSON	JavaScript Object Notation
LCS	Lecture Capturing System
MB	Megabyte
ML	Machine Learning
MOT	Multi-Object Tracking
OpenCV	Open Computer Vision
PTZ	Pan-Tilt-Zoom
RAM	Random Access Memory
ROPE	Region of Practical Equivalence

S	System
TB	Terabyte
TN	True Negative
TP	True Positive
VC	Virtual Cinematographer

1 Introduction

Lecture recording has existed for many years [8] and has evolved significantly with the advent of modern technology. Early systems used expensive equipment and manual labour [19]. *Automated Lecture Recording* (ALR), also known as Lecture Capturing System (LCS), is the semi-automatic process of capturing live lectures and making them available online for students to view in their own time. These ALR systems have also seen much development, from using expensive Pan-Tilt-Zoom (PTZ) cameras to a newer approach using inexpensive, high-resolution 4K static cameras. Earlier ALR systems utilised an overview and dedicated Pan-Tilt-Zoom (PTZ) camera to physically pan and zoom during the lecture. Although these approaches are still widely used [95], they are expensive and have limitations. As the availability of lecture recording at educational institutions continues to expand [72], the use of 4K cameras has become an affordable option to capture high-resolution videos.

A high-resolution 4K camera provides crisp images with good writing visibility on boards or projector screens. Unfortunately, the resultant video has a very large file size, with an average 45-minute lecture exceeding 2GB. This poses a limitation in low-resource environments, particularly in developing nations, since university budgets and disposable income are often severely constrained. Firstly, processing, storing and distributing these size videos for many lectures require the institution to invest in high-end expensive servers and networking equipment, posing a significant cost obstacle. Secondly, students viewing these recordings off-campus would require a fast broadband connection with a large data quota since viewing several videos a day would use a great deal of data over the span of a month. This is further exacerbated if students are using expensive mobile data bundles, which may mean that lecture video access is infeasible for many. This is the context within which this work was undertaken.

Downscaling from 4K to a lower resolution is a viable method of reducing the file size. Unfortunately, this generally reduces video clarity and defeats the initial purpose of using high-resolution cameras. Instead, a re-targeting approach using a Virtual Cinematographer (VC) identifies a region of interest in the frame and directs the viewer's attention to what is essential by extracting a smaller cropping window from the full 4K frame as illustrated in Figure 1.1. This approach retains the image quality of the video while producing a lower-resolution output video with a smaller file size.

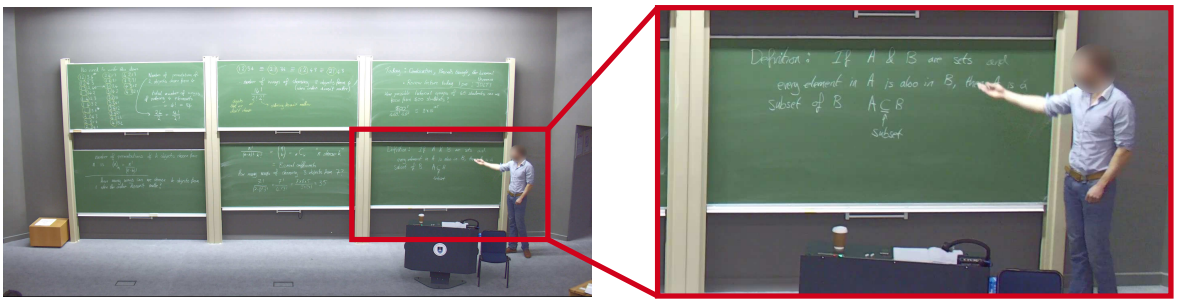


Figure 1.1.: This figure illustrates the process of segmenting a smaller 720p cropping region from the input 4K frame (re-targeting).

1.1 Aims and Approach

In this work, we explore three aspects — presenter, gesture and writing detection in pre-recorded lecture videos — to be used as a detection front-end in a larger pipeline with a VC. Our front-end aims to supply an external VC with contextual information to generate a smaller *cropping window* within the 4K frame (see Figure 1.1), providing a context-centred view of the presenter (also referred to as re-targeting).

Machine Learning (ML) based object detection approaches have recently received much attention due to their effectiveness. These approaches are, however, resource intensive and typically only provide fast runtimes when used with a Graphics Processing Unit (GPU). Due to the unconstrained nature of presenter motion, the detection algorithm needs to run frequently to avoid jittery tracking. This is especially true for gesture detection since a gesture is typically short in duration, and a low detection frequency could miss it altogether. The high detection frequency required for presenter and gesture tracking renders an ML approach for this purpose infeasible in our low-resource context.

Writing detection serves as a proxy for detecting board/screen usage regions and aims to further enhance the VC's context-centred view by including these regions that may be relevant to the presentation. Unlike presenter detection, which requires a high detection frequency, writing is typically added incrementally and remains in the same position for longer. A lower detection frequency is therefore suitable for this purpose. Furthermore, any inaccuracies in writing detection would have a less significant impact than poor presenter detection.

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 GPU. Furthermore, the algorithms are designed to support high video throughput with a limited memory footprint, thus limiting the memory requirement for processing 4K video clips.

More specifically, we investigate the feasibility of using an inexpensive image differencing algorithm for presenter and gesture detection. We investigate the feasibility of using the Efficient and Accurate Scene Text (EAST) detector with a low detection frequency to detect writing in our context. We also explore the effect of relevant environmental factors on the algorithm's outcome for all modules.

Research Questions Based on the previously discussed aims, we have identified the following research questions for each module in our system.

1. Presenter tracking
    a) How accurately can an inexpensive image differencing algorithm detect a presenter in a lecture venue in a post-processing approach?
    b) How do environmental factors such as lighting and the number of targets in view influence the tracking accuracy?
2. Writing detection
    a) How accurately can the EAST detector using the pre-trained neural network detect writing on boards in a lecture venue?
    b) How do environmental factors such as lighting, writing visibility, diagrams/images and protector text or handwriting influence the detection accuracy?
3. Gesture detection
    a) How accurately can we detect the presence and direction of gestures as an extension to the frame differencing?

1.2 Contribution

The main contribution of this research is a carefully designed front-end that performs low-cost presenter and writing detection as input to an external VC. Furthermore, a set of heuristics was developed to mitigate the limitations of this low-cost approach.

We show that our low-cost differencing and heuristics for presenter detection can be used to implement lecture tracking, as an alternative to ML-based approaches, across a range of environments. Our results showed a mean file size reduction of 81.3% and a total runtime of

less than 1.5 times the input video length when using our front-end in conjunction with an external VC.

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 degraded with students' heads in view.

1.3 Overview of Dissertation

The remaining chapters of the dissertation are structured as follows: Chapter 2 contains background information and a literature review on object, gesture and writing detection methods. Chapter 3 describes the implementation of the three core modules — presenter, gesture and writing detection — in our front-end. We present our evaluation methodology and results in Chapter 4. Finally, Chapter 5 concludes the dissertation with a summary of findings, followed by suggestions for future work.

Note: A blur filter has manually been applied to all figures containing presenters to preserve anonymity. This operation does not affect our system as it was done externally for images used in the dissertation only.

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

2 Background and Literature Review

This chapter introduces the relevant background necessary to contextualise this project, followed by a survey of the relevant literature for each module of our project.

2.1 Background

A brief history of lecture capture Motion-picture films, the early form of what is known as movies or films today, were the fundamental driving factor in the move towards visual education [34]. The use of motion pictures in the classroom for educational purposes gained momentum from the 1920s, soon followed by the audible motion picture, which included sound [26].

Early lecture capturing was limited by the available technology at the time, and recording a lecture was a dedicated and meticulous process (similar to film production) [8]. Rapid technological advancements in the 1980s made interactive distance learning possible by linking classrooms via connections such as cable, microwave or satellite, creating a "virtual classroom" [47].

During the 1990s, with the expansion of the internet and the availability of computers, the idea of a virtual classroom continued to grow and became available off-campus to the public [48]. Video/Audio recordings of the presenter and teaching materials such as the blackboard or projector transparencies were digitised and stored on a multimedia server and accessed via the internet. This advancement resulted in the idea of "Lecture on-demand" since it could be accessed anytime and repeatedly.

Later lecture capturing evolved from using a static camera without tracking [1] to using dedicated tracking camera hardware to provide a close-up shot of the presenter [61]. Switching between different camera feeds provides the viewer with a better sense of presence and is more engaging [9]. Including writing boards in the view was also recognised as being important in a lecture capturing system.

Modern visual presenter tracking approaches can be divided roughly into two approaches: *real-time* and *post-processing*. Real-time approaches (e.g. [96]) often used Pan-Tilt-Zoom (PTZ) cameras and are useful when live broadcasting is desired, such as sports events [41], meetings [31] and lectures [9].

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 [37] and is often used in conjunction with one or more high-resolution cameras to segment a smaller crop region from the input frames enclosing the presenter [21].

This concept of creating a smaller cropped view from a larger view has been used for re-targeting or resizing the video to fit specific aspect ratios [43], to smooth out PTZ tracking motor control errors [17] and creating a split screen view to show close up views of actors [53]. Liu and Gleicher [55] summarises the process of video re-targeting as preserving what is essential in the video and discarding the rest. To do this, there needs to be a method of detecting what is essential. This is where our work is situated, creating the object detection method that would enable an external virtual cinematographer to make the final cropping decisions.

Lecture Capturing Systems Today, many different environments and disciplines such as conferences, meetings, seminars and sporting events use and benefit from automated recording systems [57]. This research, however, will focus specifically on the use case of lecture recording in an academic institution.

Lecture capture, as it is known today, refers to the recording of live lectures as an additional learning resource [30]. Lecture capturing enhances the students' experience [83] and considering that students are the "clients" of lecture capturing [65], the design of such a system should aim to provide output videos that reduce the cognitive load of the learner and direct the viewer's attention to what is essential [32].

Brooks et al. [13] describe the components of a typical automated video capturing system. These systems often allow integration with other platforms or subsystems that can work together to provide the overall automated video recording functionality to meet the custom requirements. Such functionality includes capturing the event at a predetermined schedule, processing the captured video and making it available for a particular audience on a distribution platform often referred to as a Learning Management System (LMS) or Virtual Learning Environment (VLE). The architecture or workflow of automated lecture capture systems follows a similar design pattern and consists of the following high-level components.

Content acquisition The content acquisition subsystem interfaces with the physical capture device that combines the video from the cameras and audio from the microphones in the classroom. Video files are a collection of sequential images, referred to as frames [92]. The frame rate represents the number of frames per second (FPS), and to reduce the video file size, frames are compressed according to the coding standard (codec) used. The lighting conditions can also impact the tracking and should be considered when evaluating tracking systems [35].

Processing The processing subsystem is a combination of software and hardware used to process the videos and audio received from the acquisition subsystem. The software typically runs on a server and allows scheduling recordings, managing recorded videos and publishing completed videos to a distribution platform. The system also allows videos to be post-processed, which is where our work is situated.

Distribution The distribution subsystem ensures that the end users can view the final processed video — the software interfaces with learning management systems that allow enrolled students to access the videos.

2.1.1 Presenter Detection

Many different tracking techniques exist depending on the type of sensor used [56]. However, our focus is on vision-based tracking techniques capable of detecting a presenter in a video stream to inform an external virtual cinematographer module of the presenter's position.

Presenter detection (a subset of object detection) and tracking are used across various applications, including corporate meetings, television, lectures and presentations for broadcasting or recording for later viewing [91]. The detection and tracking of presenters are helpful in focusing the viewer's attention on what is essential, as well as capturing close-up shots that show the presenter's facial expressions, gestures and any writing or text on presentation media such as projector screens, whiteboards and blackboards.

Object detection and tracking form a fundamental part of computer vision that has been extensively studied and remains a very current area of research. In recent years, Multi-Object Tracking (MOT) has seen much growth due to an ever-increasing number of applications, such as autonomous driving, that benefit from their ability to detect and track multiple objects [90]. Furthermore, they can perform the detections in videos with non-static backgrounds, which significantly increases the tracking complexity [28]. These approaches often utilise machine learning and require a GPU or high processing power to reach the stated runtimes. Instead, we

focus on classical (non-machine learning) object detection approaches that do not have these requirements.

Classical object detection approaches aim to separate the foreground objects from the background and typically begin with the critical process of detecting motion between two video frames by analysing the change in pixels between these frames [40]. Background subtraction, temporal/frame differencing, and optical flow are three prominent techniques used to perform object detection [79].

Background subtraction [14] is a widely used and efficient technique available for determining motion. The process starts by building a background model (or reference frame) from an initial frame. Each frame in the video is then subtracted from the reference frame, pixel-wise, and then thresholded, resulting in a binary image which can then be subjected to morphological operations to extract the foreground objects. The background model is constantly updated to accurately depict the current background state since objects previously considered part of the background may have moved away. This model maintenance generates additional computational costs to this approach. The algorithm is only as good as the background model [84] and is very sensitive to illumination changes and dynamic (busy) scenes. These constraints are still receiving attention in recent research [76].

Temporal differencing [73] is a fast and low computational cost method that is similar to background subtraction, however, two successive frames (e.g. 1,5) are now being subtracted, pixel-wise. This is also known as frame differencing when two adjacent frames (e.g. 1,2) are subtracted. Compared to background subtraction, this technique is more adaptive to dynamic backgrounds and illumination changes. The major disadvantages of this approach are that it does not extract the complete shape of the moving object, but leaves behind holes [29] and does not detect stationary objects. Shadows can also distort the object or be detected as a separate object. Furthermore, foreground ghosting occurs when the object moves too fast and/or the frame rate is too low, creating two instances of the same object, while foreground aperture occurs when the object moves too slowly, leaving only a thin outline of the object.

The foreground aperture and ghosting issues introduced by temporal differencing can be solved using a three-frame differencing technique [97], slightly increasing the complexity. This works by finding the difference between frames n_{-1} and n and frames n and n_{+1}. This technique requires additional differencing operations. The total number of differencing operations for n frames using the two-frame differencing technique is $n - 1$ and for the three-frame approach is $2n - 2$, essentially twice the number of differencing operations.

Optical flow [29] uses an optical flow field to detect motion by analysing and matching the position of pixels over frames. A vector field is built to indicate the velocity and direction of these

pixels and an object can only be detected if its velocity is distinguishable from the background. Discontinuities in this motion field can then be used to segment regions representing the objects of interest. One of the major disadvantages is that the optical flow calculation is complex [7] and sensitive to noise. From the approaches discussed, it is computationally the most expensive option.

Table 2.1.: Comparison of the commonly used classical motion detection techniques.

Method	Accuracy	Runtime	Limitations
Background subtraction	Moderate	Moderate	• Dynamic backgrounds • Illumination variations • Requires an accurate background model
Temporal differencing	Low	Fast	• Foreground aperture and ghosting • Low or no motion (stationary objects) • Produces jittery motion
Optical flow	High	Slow	• Computationally complex • Sensitive to noise • Illumination variations

Table 2.1 summarises the key differences between these three techniques. Optical flow provides the best accuracy, however, it is computationally the most expensive. Temporal differencing is faster than background subtraction [73] and is suitable for scenarios with changing backgrounds, which is required in our project since boards and screens can move.

Thresholding is a fundamental step used to segment the foreground objects from the background. The pixel values in a differenced image are analysed and compared to a pre-defined threshold. Pixel values greater than the threshold are set to 255 (white), while pixel values less than the threshold are set to 0 (black). This process creates a binary (black and white) image with white pixels depicting regions of motion between the input frames [2]. These thresholding techniques can roughly be categorised into two groups, global and local [46]. Global thresholding uses a single threshold value for all pixels in the entire frame, while local thresholding uses a unique threshold value for each pixel [33].

Global thresholding techniques are very popular and commonly used for their simplicity and effectiveness [46] and have a lower computational cost. Otsu's method [67] is a popular global technique that calculates a single threshold value for all pixels. Two commonly used local thresholding techniques are the algorithms of Niblack [63] and Sauvola and Pietikäinen [75]. In the work of Pai et al. [69], a comparison is made between these algorithms and Otsu's method is shown to have the lowest computational complexity and runtime. Figure 2.1 shows a comparison of Otsu's (global), Niblack's (local) and Sauvola's (local) algorithms for three

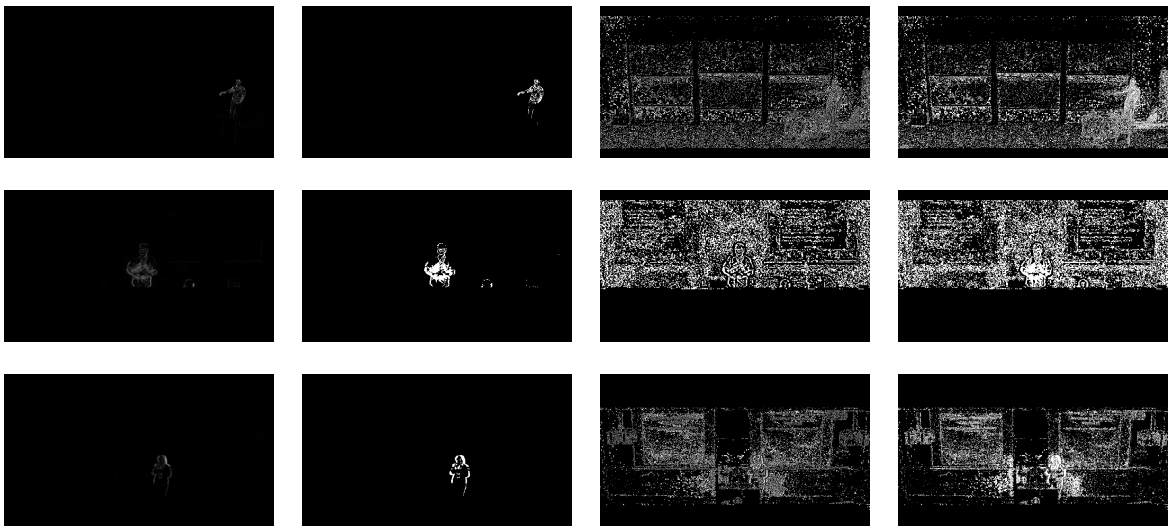


Figure 2.1.: This figure shows the difference between Otsu's thresholding algorithm (global) and two local thresholding algorithms (Niblack and Sauvola) for three different illumination levels ranging from good (top row) to poor (bottom row).

lighting scenarios that are typically encountered in lectures. These are summarised into three lighting levels: good (top row), moderate (middle row) and poor (bottom row). Otsu's method provided better results across all three lighting scenarios. These images were produced using the auto thresholding functionality in the Fiji [1] distribution of the ImageJ software [77].

Post-processing techniques such as morphological operators, connected components analysis and flood fill algorithms are typically used after thresholding the image to remove noise and isolate the silhouettes [79]. After the image has been improved, detected objects can be represented by one of the many options [101] illustrated in Figure 2.2. In most applications, rectangle bounding boxes are sufficient for enclosing detected objects.

In summary, the minimalistic set of algorithms required to detect objects (motion) using the temporal differencing approach is outlined below.

Figure 2.3 shows the core components of an object detection implementation using temporal differencing. First, a pixel-wise difference (as discussed above) between two input frames is computed. The output frame (or image) from this stage is a greyscale image with each pixel representing the difference between the same pixel in the two input images. The greyscale image produced during the differencing stage is passed through a thresholding function to create a binary (black and white) image that represents the foreground (white) and background (black). At this stage in the pipeline, small foreground pixels often known as noise are present

[1] https://imagej.net/software/fiji/

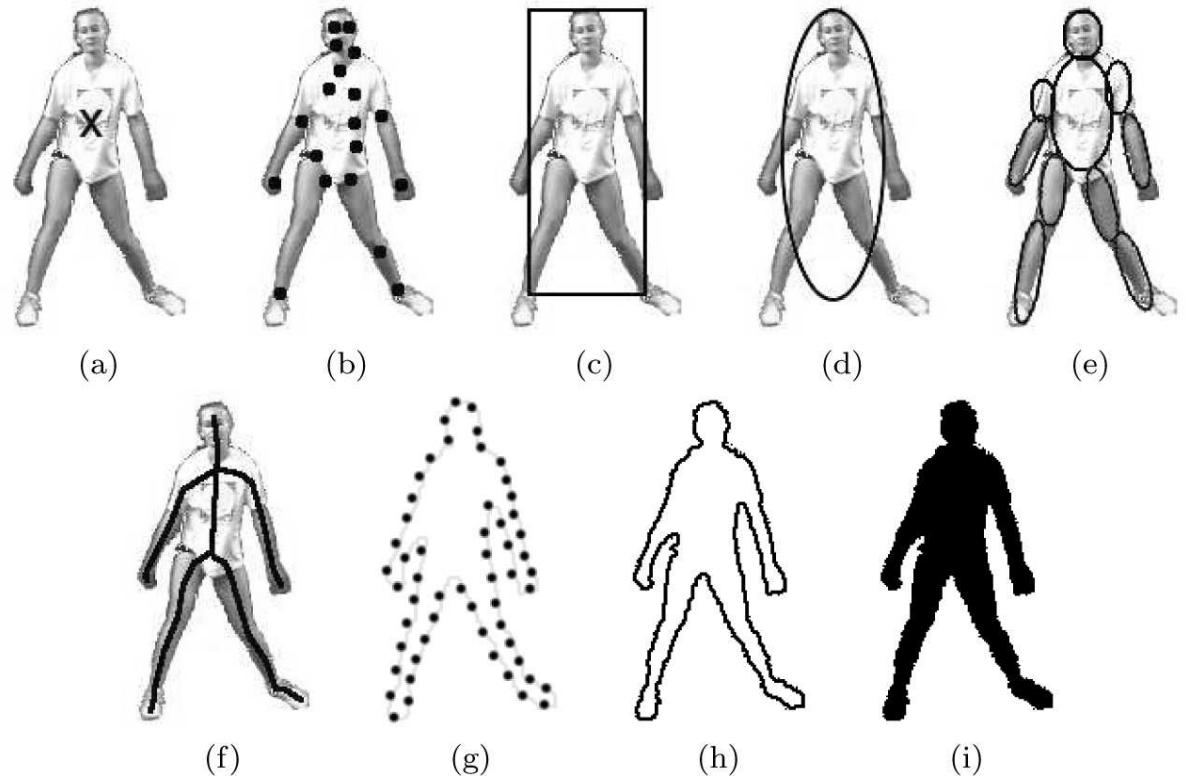


Figure 2.2.: This figure is from the work of Yilmaz et al. [101] illustrating various object representations. Rectangle bounding boxes (c) are commonly used to represent detected objects.

in the binary image (see for example Chapter 3, Figure 3.5). Morphological operations are used to remove this noise followed by a connected components analysis that groups the remaining connected pixels into "blobs" (groups of pixels that changed between frames) representing the final detected objects in the frame [78]. Lastly, a tracking stage uses these detected objects to establish their persistence between frames.

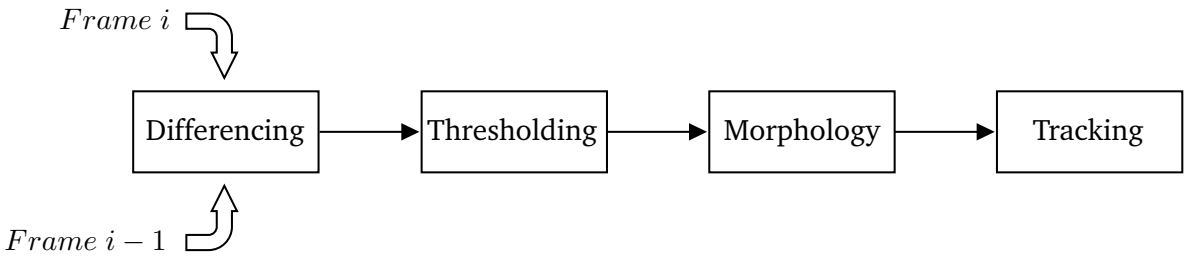


Figure 2.3.: High-level flow diagram of frame differencing.

2.1.2 Gesture Detection

Gestures are an essential form of non-verbal communication and a diverse area of research that pertains to the recognition of human motion involving the arms, face, hands, head and body [71].

Gesture information can be used to guide the camera to capture more information and provide context [103] essentially working to achieve the ingredients proposed by Lampi et al. [54] that are required to make a good lecture recording system. Wang et al. [88] also use gestures and writing/text as supplementary information to enhance the manual editing process of lecture recording videos.

We explore gesture detection as a subset of presenter tracking because once the presenter's position is known, it is easier to look for gestures. We focus on exploring the detection of large pointing gestures as an extension of presenter detection.

2.1.3 Writing Detection

Writing detection in the context of this project is done purely as a proxy for finding board/screen usage regions and is aimed at providing additional contextual information to assist and improve the framing decisions of an external VC.

Many applications can benefit from knowing the location of writing on a board in a lecture, such as summarising a lecture [98], guiding the viewer's attention to what is essential [37] and improving search and navigation [81].

Writing detection is a very broad area of research in both image and video applications. However, detecting scene text in videos is considered more challenging and recent research trends have effectively used machine learning approaches to solve some of these challenges [100]. Detecting and segmenting text in a lecture video is challenging because of the different types of text that could be present. This could be text on a projector screen from a presentation or a document camera which could contain characters that are either handwritten or a font. Text is also written on chalkboards or whiteboards, which has many challenges because of handwriting style, size of the writing or the contrast of the writing on the background surface.

Figure 2.4 shows a range of venues considered in this work, and writing detection should not make assumptions about the venue layout.

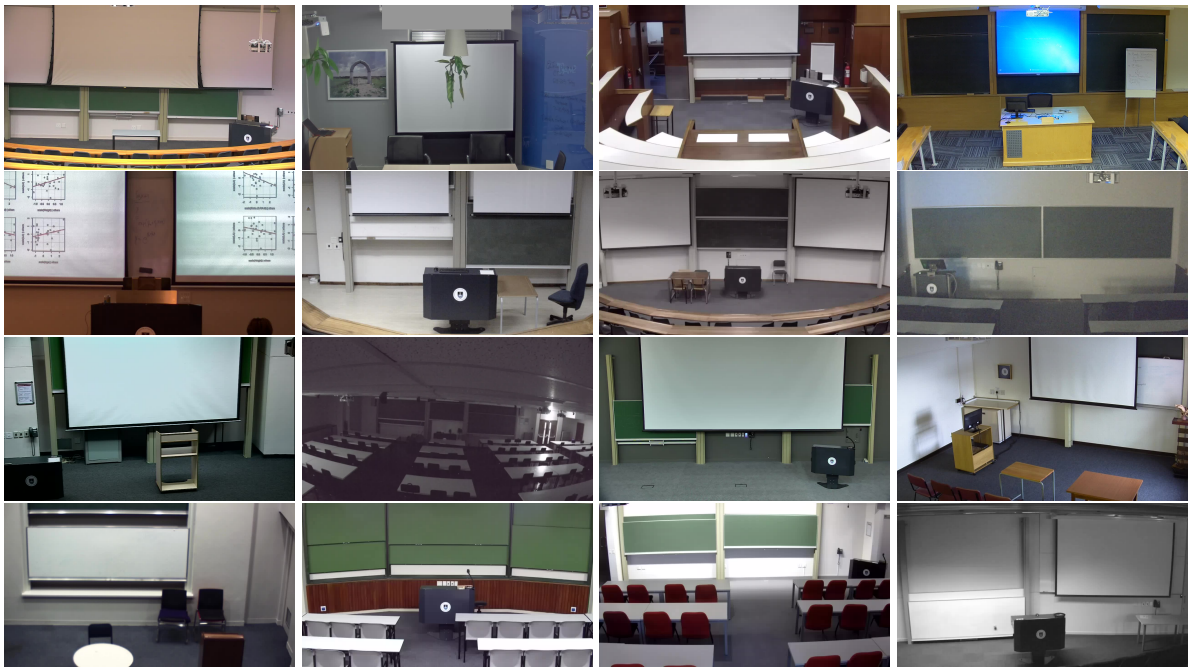


Figure 2.4.: Images showing a range of venue layouts considered in this work.

2.2 Literature Review

In this section, we provide a survey of relevant literature for each module in our project.

2.2.1 Presenter Detection

The significant success of deep learning (DL) and other machine-learning-based approaches [16] has led researchers to focus on exploring these state-of-the-art techniques [94]. Unfortunately, deep learning applications typically depend on GPU acceleration to obtain the full speed benefits for inference and learning [24]. For example (see Figure 2.5), in the pose estimation system presented by Cao et al. [16], inference time using the GPU is $36ms$ compared to $10396ms$ for the CPU-only version. This method would provide precise tracking and gesture information, however, it is infeasible under our hardware constraints.

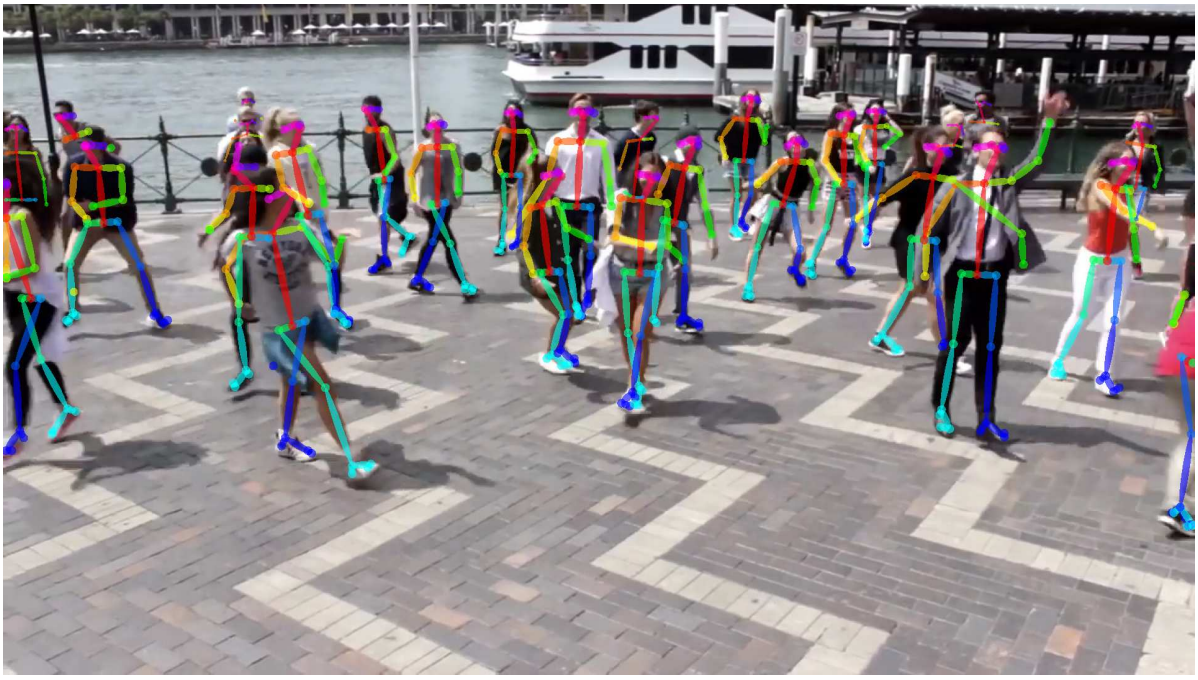


Figure 2.5.: An image from the work of Cao et al. [16] showing the output of their pose estimation approach.

While writing detection can be run infrequently (every n^{th} frame, for small n), presenter detection requires a higher detection frequency to cope with a moving presenter. This means a DL approach is infeasible for our context if used without a GPU.

In contrast, classical motion detection approaches [52] do not require high-performance hardware or GPU acceleration and are often used on embedded processors of cameras which

have limited processing power [104]. Video analysis has three main phases: motion (object) detection, object tracking and object analysis [52].

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 [73, 80]. Additional object classification and tracking stages [5] add further computational costs and are not essential in our context — since there is typically only a single presenter in view.

Optical flow provides the best accuracy but is computationally the most expensive. Temporal differencing is faster than background subtraction [73] and is suitable for scenarios with changing backgrounds, which is required in our project since boards and screens can move. Temporal template approaches [23, 105] provide better silhouettes than temporal differencing but are computationally more expensive since all frames are analysed.

In the approach of Mavlankar et al. [60], motion, skin colour and background subtraction are used to track the presenter across frames. Using a fixed skin colour is problematic due to varying lighting conditions and natural skin colour variability. They restrict the tracking to one dimension on the observation that the lecturer's motion is predominantly horizontal. This poses a problem when porting the solution to a venue where vertical motion is possible.

Yokoi and Fujiyoshi [102] and Arseneau and Cooperstock [3] 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.

Since we do not require the precise presenter location but rather a region of interest, temporal differencing is suitable for detecting motion [103] in our context. In our context, there is typically only one presenter in the camera field of view and excluding a tracking module as Yokoi and Fujiyoshi [102] does will further reduce computational costs.

2.2.2 Gesture Detection

Hidden Markov Model (HMM) based classifiers are commonly used for gesture recognition in human-robot interaction applications [64, 70] while others are used in the classroom for detecting either hand-raising gestures by students [39], detecting presenter gestures [89] or handwriting gestures [59]. HMMs require extensive training, and, although attempts have

been made to minimise these training requirements Natarajan and Nevatia [62], this technique does not align with our constraints and the differencing-based approach for presenter detection. Some approaches [49, 10] detect gestures by first locating the face using a face detector followed by searching predefined sub-regions based on the face region as seen in Figure 2.6.

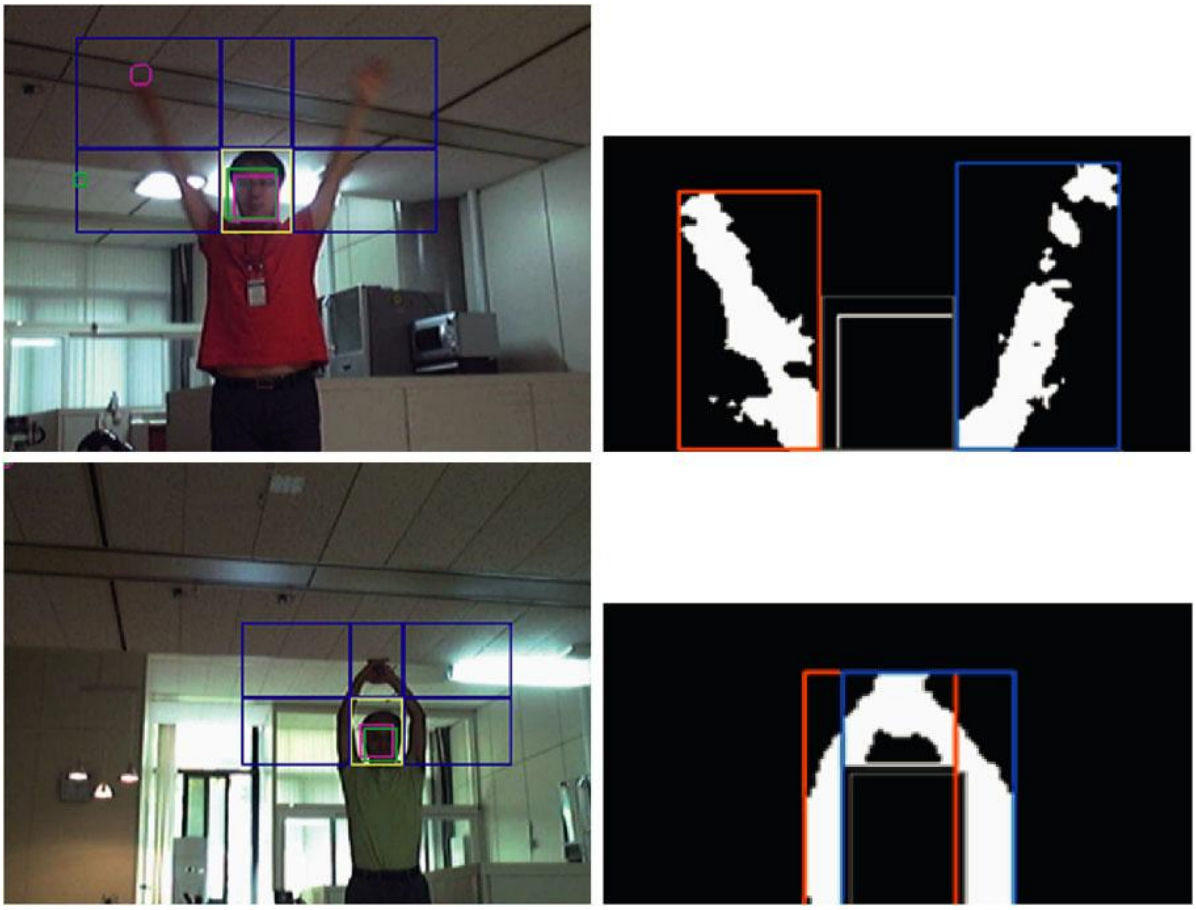


Figure 2.6.: An image from the work of Kim et al. [49] showing how they detect gestures in set regions around the head after first identifying the face of the person.

Yao and Cooperstock [99] use frame differencing and morphological operations followed by edge detection and skin tone to detect the gesture. Using skin colour is not recommended since there is no universal skin tone [3]. Wulff and Fecke [95] proposed detecting presenter gestures in future work by using image differencing and evaluating the distance from the outermost pixels to the object centre. A large distance would likely represent a pointing gesture. We explore this approach since it aligns well with our efficiency aims and can be added as an extension to object detection.

2.2.3 Writing Detection

Colour segmentation is widely used [18] and early approaches [38, 36, 87] used pre-defined colours to identify the chalkboard regions. These approaches degrade under changing environmental factors such as moving boards, lighting changes, and non-uniform board colour due to wear and remaining chalk dust. Using fixed colours may also require recalibration when used in different venues with different colour boards or projector screens.

Adaptive colour selection approaches [20, 25] used the average colour from the largest regions to segment the boards. This method relies on the assumption that the boards are typically the largest region in view. Assuming the board is the largest consistently coloured region has still been used recently [22] and seems a fair assumption. Unfortunately, different venues or camera configurations may lead to instances where the boards/screens are not necessarily the largest consistently coloured region(s), resulting in the wrong segmentation colour being selected.

Some approaches [42, 82] use motion detection or feature detectors [6] to first detect candidate text regions followed by a classifier. One drawback of these approaches is training the classifier to isolate the text correctly in the pre-processing stage.

Yadav et al. [98] proposes a method that, instead of focusing on text, attempts to detect "anchor points" such as figures, tables, equations, flowcharts, code snippets and charts using a Convolutional Neural Network (CNN) that is trained using a database of images. The system, however, only works with slide-based videos and using it with blackboard or handwritten videos results in diminished accuracy. These limitations restrict the application to videos that only use presentation slides, which is not useful in our context.

An Efficient and Accurate Scene Text Detector (EAST) proposed by Zhou et al. [106] uses a Deep Neural Network (DNN) with only two stages to detect text, which improves detection speed. The pre-trained model was trained using the International Conference on Document Analysis and Recognition (ICDAR) 2013 [44] and 2015 [45] training datasets. Unlike the other approaches discussed thus far, the efficacy of the EAST method was not demonstrated in a classroom setting with handwriting and mathematical equations. The authors claim it can robustly detect various forms of text in challenging scenarios such as non-uniform lighting, low resolutions and orientation variations.

Unlike presenter detection, writing detection can be run at a lower frame rate [20], enabling the exploration of a more expensive approach at a lower detection frequency. A foreseen limitation of EAST with the authors' pre-trained model is the ability to detect diagrams and images, however, we assume that in typical use cases, there should be some text in the scene.

At the time this component was developed, EAST was considered state-of-the-art, and we adopted it as our text detector. However, newer methods have since been developed, for example, the Character Region Awareness For Text detection (CRAFT) [4] and the work of Urala Kota et al. [85], which generally improve robustness. This should readily fit into our pipeline architecture, but that is left for future work.

2.3 Summary

This chapter introduced relevant background information about lecture recording to contextualise the project, followed by a presentation of relevant literature in the three fields explored by our project — presenter, gesture and writing detection.

Lecture capture systems are used to capture, process and distribute lecture recordings. A re-targeting approach is explored in this project and requires the presenter's location and a classical temporal differencing approach was chosen as a viable option to explore in the context of our project. We also discussed the rationale for including gesture and writing detection as supplementary contextual information. A gesture detection algorithm using the pixel distribution was chosen since it can extend the frame differencing approach. Although machine-learning-based approaches are hugely successful, they are resource intensive and infeasible for presenter and gesture detection in our low-resource context.

Lastly, a state-of-the-art machine-learning-based writing detector, EAST, was chosen due to its notable robustness. Like other ML approaches, the EAST method requires a GPU to obtain a fast runtime. While presenter and gesture detection requires a higher detection frequency to avoid jittery motion, writing detection can accommodate a significantly lower detection rate. This is due to the assumption that writing is added incrementally over time and remains in the same location for extended periods. Using EAST at a lower detection rate without a GPU allows it to remain a feasible option to explore in our low-resource context.

In the following chapter, we discuss the design and implementation details of these three modules in our project.

3 Video Analysis Framework

3.1 Overview

This chapter describes the video analysis framework, comprising the presenter, gesture and writing detection modules. The framework includes additional helper components such as file readers and writers for the video, configuration and output files required to implement the core functionality. All algorithms are clearly explained, and brief discussions on implementation details are included where necessary.

The chapter is divided into seven sections. The first two sections discuss the system design and architecture. The subsequent sections discuss the implementation details for the presenter, gesture and writing detection, respectively, followed by the data filter module. The last section discusses the programming language and libraries used for the project.

3.2 Design Goals

These requirements are informed by the LCS implementation at the University of Cape Town, which has a large number of challenging lecture venue configurations. The data used for this work was sourced from the same institution and is described later. As stated in Chapter 1, the system design should aim to maximise processing speed on general-purpose commodity equipment without using a GPU, and the processing turnaround time should be minimised to avoid backlogs of unprocessed lecture recordings. This is achieved by using cheap image differencing-based techniques followed by heuristics to mitigate the shortcomings of this approach. Furthermore, the solution should be fit for purpose (i.e. should produce output which is sufficiently good for the purpose of automated lecture recordings) 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 set-up, parameters should be self-calibrating as much as possible and additional parameters that are configurable should be stored in a central configuration file. The code of this project will also be made open-source to enable further developments and usage.

3.3 Core Design

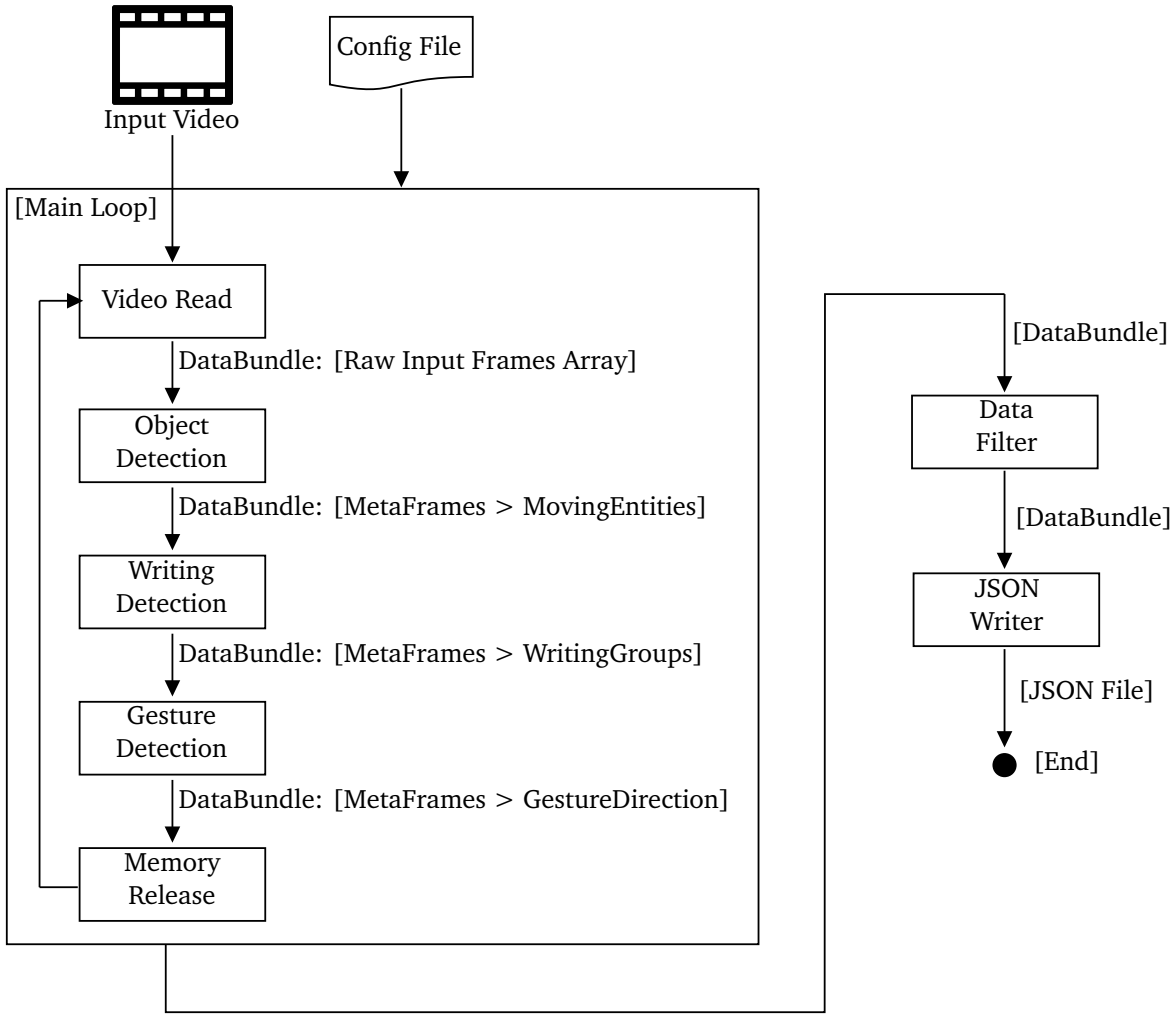


Figure 3.1.: Overview of the complete system architecture.

Figure 3.1 shows the high-level flow of data through our system modules. Appendix A provides the pseudocode of the model showing the flow of data through the system. To facilitate this flow of data, 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 are required by later modules in the pipeline. The videos are processed in chunks to reduce the need for large amounts of memory and allow any reasonable 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.

Using the pipe and filter architecture allows the input video to be processed by each module while the resultant data is retained and passed down the pipeline leaving the final output with all the data required to evaluate the system and answer the research questions posed in Chapter 1. The system was implemented using an Object Oriented approach to keep all components and data organised and modular. Together, all these components form a complete presenter, gesture and writing detection system that can be used with an external VC to create a complete post-processing module in the LCS pipeline.

A brief overview of the modules and containers shown in Figure 3.1 is discussed below. The three core modules — **Motion detection**, **Gesture detection** and **Writing detection** — are discussed further in their own dedicated sections.

Configuration file The configuration file is a central location, external from the program, to save default parameter values. It also enables the end user to customise and calibrate the parameters for different venues without recompiling the source code. The JavaScript Object Notation (JSON) is human readable and a commonly used data-interchange format and was chosen as the file format for the configuration file. Appendix D shows the file format of this configuration file along with the system parameter names and their purpose. Four of these parameters (*skipFrames*, *workingDimension*, *writingDetectionSkipTime* and *writingConfThreshold*) and their purpose are discussed in more detail throughout this chapter.

Video file reader This module reads and decodes the frames from the input video file into memory. It is also responsible for resizing the frames to reduce the processing times required by later modules. To increase processing speed, possibly at the expense of accuracy, a set number of frames can be skipped using a *skipFrames* parameter. The video file reader also stores the video file meta-data, such as the Frames per Second (FPS) and the total number of frames in the file.

Video processing loop The file size of a typical 4K lecture video varies depending on many factors, including frame rate, resolution, bit rate and the codec used. In the dataset of lecture recordings used in this project, a 50-minute 4K resolution lecture video (30fps with a maximum bit rate of 4Mbps using the H264 codec) has an average file size of 2.24GB. This size at first appears to fit into RAM, however, once the video is decoded (to retrieve the individual frames), each frame was, on average, 24.9MB in size. Decoding all frames for analysis would inflate this to an unacceptable 2.7TB of memory, prompting the design choice of requiring a processing loop. A small subset (a segment or chunk) of decoded frames is retained in memory and sequentially processed, allowing any reasonable

length video to be processed. This segment size is specified using the *maxFrames* system parameter.

Once all the data has been extracted, the image data (frames) are released from memory at the end of the loop to allow reading a new segment of the video. This process repeats until all frames in the input video file have been processed.

Data container A pipe and filter architecture uses a container to carry all the data through the pipeline. This container (called *DataBundle* in our implementation) is passed through all the filters and provides the input data and configuration parameters required by each module, and stores the output data produced by each module.

The *DataBundle* contains helper functions to add and retrieve the data and the functionality to serialise the data to the JSON format, which is used by the JSON writer module. All the configuration parameters are also stored in the *DataBundle* so that they can be accessed by the other modules in the pipeline.

Meta frame To keep all the data organised in the data container, additional sub-objects were created. The *MetaFrame* object stores raw input video frames along with other information such as the timestamps in the original video and the index (frame number) after decoding. It also contains empty objects that act like placeholders to store data that is generated by the modules (filters). Lastly, it contains helper functions that interface with other modules, export their data and manage their memory usage.

Data filter This module applies a set of heuristics to the extracted data to remove outlying motion that has a low probability of being the presenter, repair missing data and refine the overall output. Further discussion on this module can be found in the implementation details for the presenter detection module (Section 3.4).

JSON file writer We chose to export all the data in the JSON file format. All presenter, gesture and writing data from the *MetaFrame* objects stored in the *DataBundle* are serialised into the output file. Additional data are also exported into the file, and a more detailed breakdown of the output file can be found in Appendix D.

3.4 Presenter Detection

Design After surveying the literature as discussed in Chapter 2, Section 2.2.1, temporal frame differencing was identified as a very low-cost motion detection method. Many object detection applications require a tight bounding box, however, for our purpose, we do not require this

precision. For this research, we explore the feasibility of using temporal differencing to detect the presenter motion in our context of driving a VC. The computational resource requirements of temporal differencing are very low, and it handles dynamic backgrounds and illumination variations robustly compared to other classical object detection approaches (see, for example, Ojha and Sakhare [66]). To reduce the effect of foreground aperture and ghosting (two common drawbacks of this method) and improve the output, a combination of morphological operations, connected components analysis and post-processing heuristics are used to improve detection results.

The detected motion is passed through a set of filters (see section 3.7) that use the mean and standard deviation of the data to define thresholds to remove outlying object detections. We assume that there is typically only one presenter in the camera view (similar to Yokoi and Fujiyoshi [102]), and when there is more, we label the frame as having multiple presenters. This labelling would signal to an external VC module that there is no specific target to focus on.

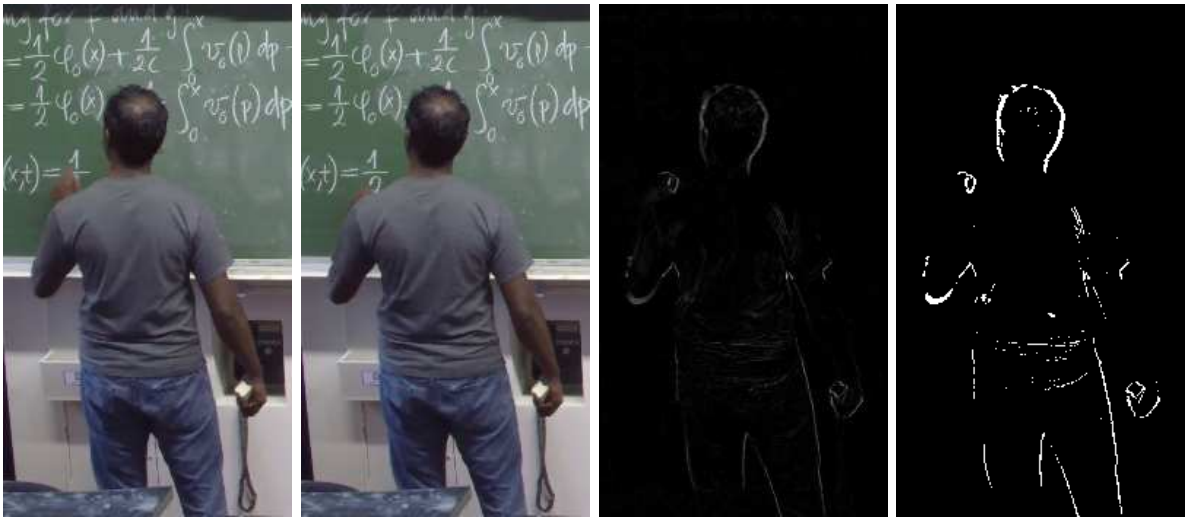
Detecting motion A pixel-wise difference is computed between two frames i (A) and $i + (\textit{skipFrames} + 1)$ (B). We defined a *skipFrames* parameter used to ignore or skip frames between frames A and B as an optimisation and a means of reducing foreground aperture. For example, if $\textit{skipFrames} = 4$, frames 0 and 5 are differenced, and $1 - 4$ are discarded. Figure 3.2 shows the output from the differencing of input frames A and B. Our detection pipeline does not require the full resolution frame, and we introduce a second parameter, *workingDimension*, which reduces the input frame resolution to the size specified by this parameter. This resizing operation is fast and will significantly reduce the processing time of the other operations in our detection pipeline.

All configurable parameters have been initialised to optimal defaults through experimentation, and details can be found in Chapter 4, Section 4.5.

Greyscale conversion A colour image would require the differencing of all three colour channels for each pixel, however, if the image is first converted to greyscale, each pixel only contains a single luminance value and simplifies the computation slightly. The small additional overhead of first converting from RGB to greyscale is offset even further since later operations only require greyscale frames. Greyscale frames also require only a third of the space compared to colour frames.

The greyscale conversion uses the following linear approximation formula to calculate the luminance value of each pixel [74]:

$$Y \leftarrow (0.299 \cdot R) + (0.587 \cdot G) + (0.114 \cdot B) \tag{3.1}$$



(a) Frame A (b) Frame B (c) Difference (d) Threshold

Figure 3.2.: Figures (a) and (b) show two input colour frames followed by (c) the difference between these frames and lastly (d) the output after thresholding resulting in a binary image where white pixels indicate change between the input frames.

Thresholding To segment the background from the foreground, a threshold function compares each pixel value to a set threshold and updates each pixel value to either 0 (black) or 255 (white), as seen in Figure 3.2d. We use the thresholding method of Otsu [67] since it has a lower computational cost when compared to adaptive threshold algorithms. Otsu's method calculates a single threshold value that is applied to all pixels in the image while adaptive methods calculate separate threshold values for individual regions or pixels in the image.

We chose three lighting categories, low, moderate and good, to represent the common illumination scenarios seen across all videos in our dataset. We selected three scenes randomly (one for each illumination level) and computed the threshold using Otsu's method and two adaptive threshold functions (mean and Gaussian) included in the OpenCV library. Figure 3.3 shows a comparison of the output for each of these thresholding techniques, and in each lighting scenario, Otsu's method produced better results.

Similarly, Figure 3.4 shows a comparison of the threshold output, for each of these thresholding techniques, on scenes with dynamic backgrounds. In this example, the presenter moves the

blackboard, resulting in large areas of motion detected by the differencing. In both figures, the noise was cleaned using a morphological opening operation using three iterations and a 3x3 square structuring element. Again, Otsu's method produced better results.

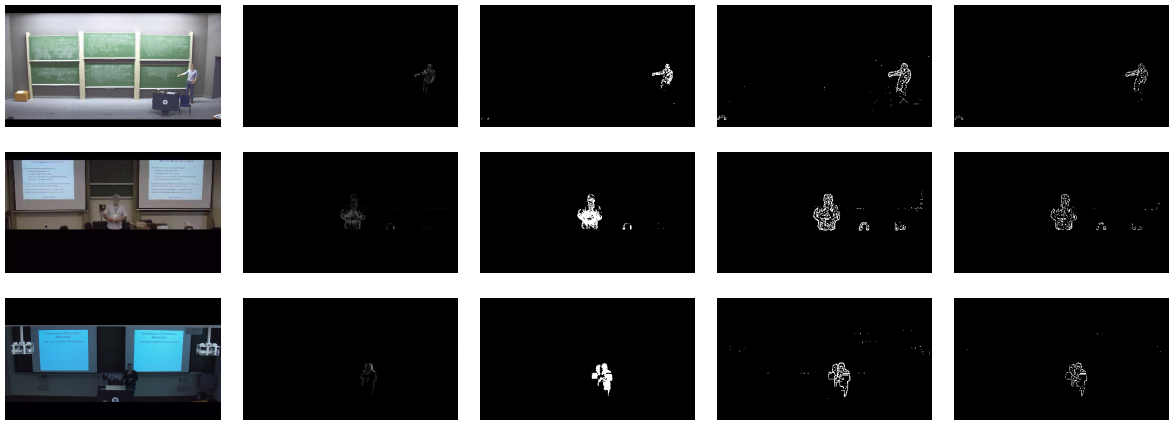
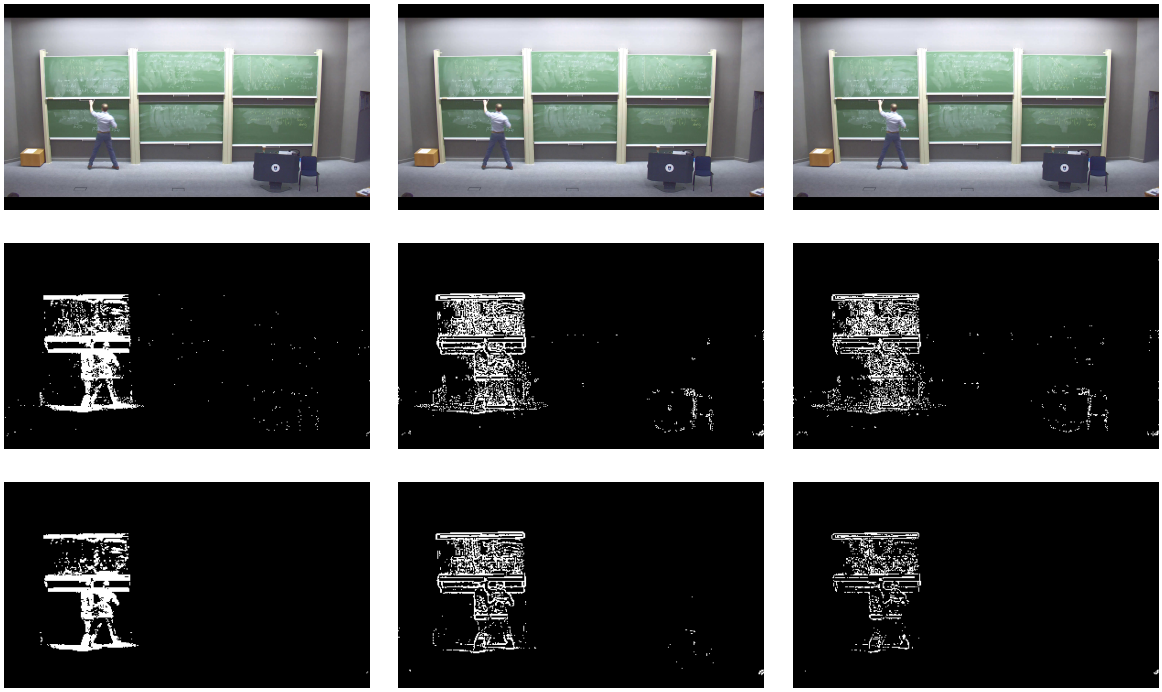


Figure 3.3.: This figure compares the output from Otsu's threshold and two adaptive threshold algorithms for three illumination levels ranging from good (top row) to poor (bottom row). These scenes have static backgrounds, and all these output images were cleaned using a morphological opening operation with three iterations and a 3x3 square kernel.

Morphology The thresholding process leaves noise (slight pixel variations between the two input frames) in the resultant image and requires cleaning to suppress the noise, as shown in Figure 3.5. Three iterations of a morphological opening operation using a 3x3 rectangle structuring element were experimentally determined to produce the best overall results across our training set of videos which cover a range of venues and lighting conditions, with examples illustrated in Figure 3.3. These morphology operations are applied to all frames using the effective parameters discussed above.

At this stage in the process, the silhouette is not solid and contains holes (foreground aperture) due to colour similarities between the two frames that result in a zero difference. Other methods, such as a flood-fill algorithm, can improve bounded regions; however, this generates additional computation costs.

Connected components analysis A connected components analysis (CCA) using an 8-connected region is then performed to isolate the blobs of pixels in the binary image corresponding to candidate objects. This results in object separation, as shown in Figure 3.6.



Otsu Mean Gaussian

Figure 3.4.: This figure compares the difference between Otsu's threshold and two adaptive threshold algorithms for a dynamic background (moving board) scene. The top row shows the colour image of frame A. The middle row shows the image before morphology operations and the cleaned images are shown in the bottom row for each algorithm.

(a) Before morphology **(b)** After morphology

Figure 3.5.: Figure (a) illustrates the noise produced by the threshold function and (b) cleaned using opening morphology operation.

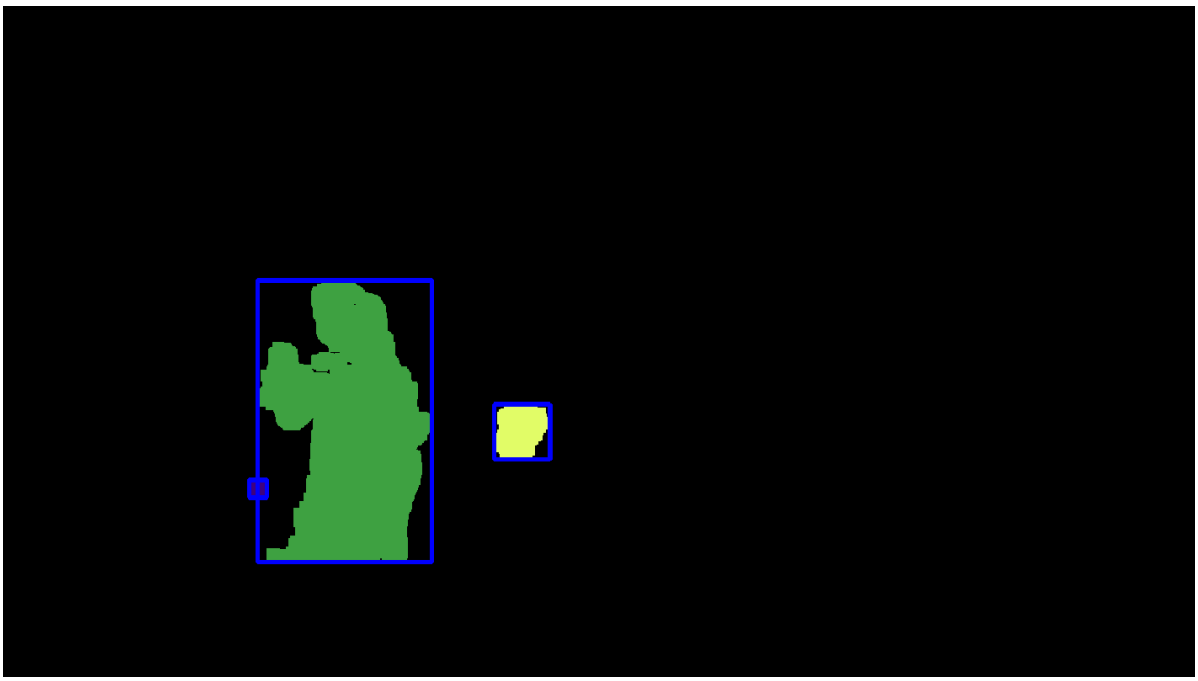


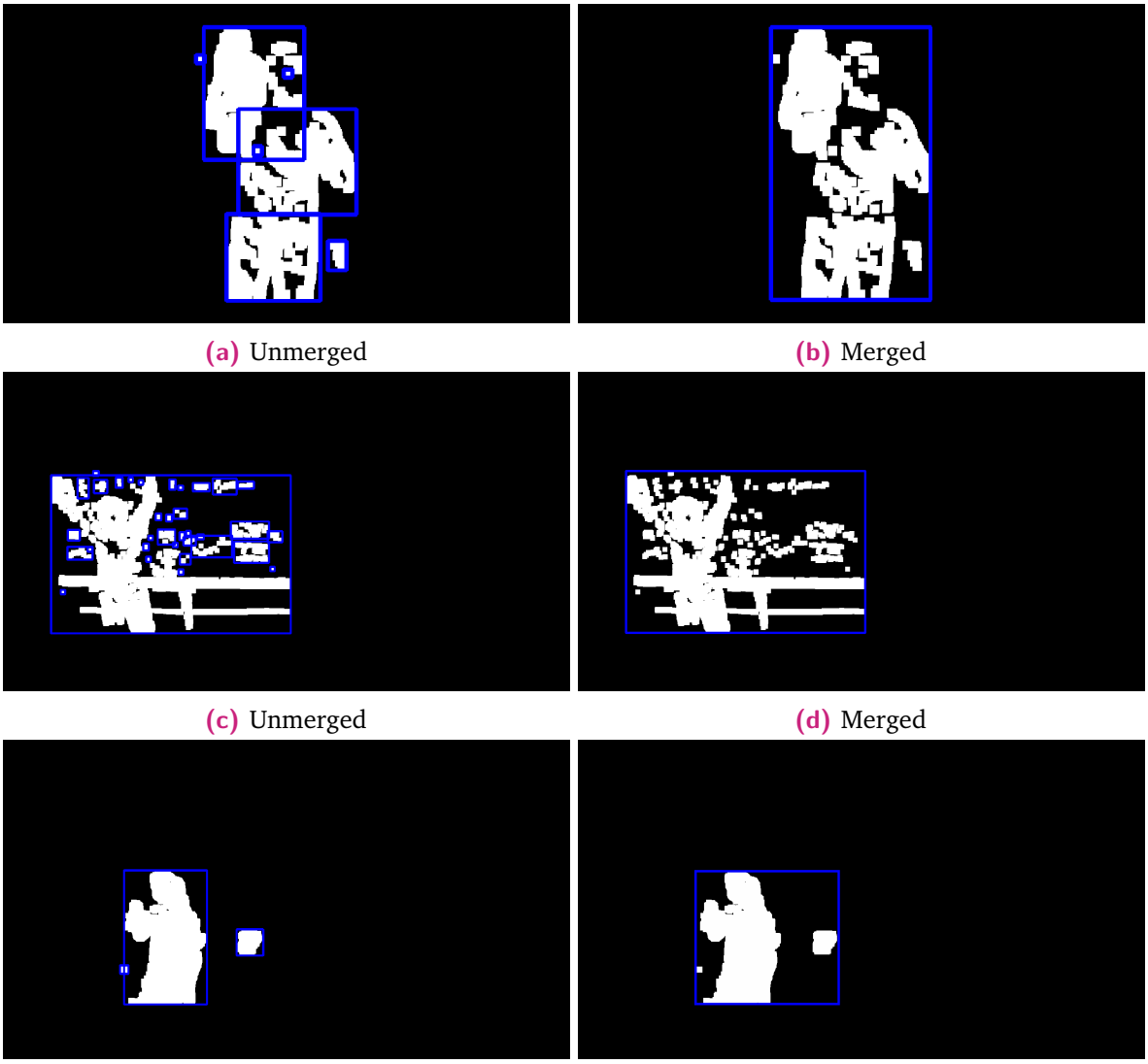
Figure 3.6.: Connected component analysis showing three separate blobs with different colours. The green region represents the presenter, while the presenter's shadow causes the other two blobs. The bounding boxes shown here are created after performing the CCA.

Bounding box creation Given that the context of this project does not require high positional accuracy, an accurate boundary representation is unnecessary, and an object rectangle bounding box should suffice. The bounding box specifies the position and dimensions of a blob in the image. The rectangular shape also simplifies other computations, such as determining the degree of box overlap.

We create a bounding box for each blob, as shown in Figure 3.6, by finding the minimum and maximum x and y values.

Clustering bounding boxes The differencing, morphology and connected component operations do not produce perfect silhouettes of the object. As discussed previously, this is a limitation of temporal differencing and is caused when there is little change between two frames (i.e. presenter is stationary) or the foreground and background colours are similar (i.e. presenter clothing colour or low lighting conditions). This limitation in pixel-wise differencing can cause fragmentation of the object into smaller regions or blobs (groups of pixels that changed between frames) that are then individually enclosed in bounding boxes, incorrectly creating separate objects. However, by clustering these object bounding boxes recursively using the distance between their vertices and a threshold, many of these fragmentation cases can be resolved, as shown in Figure 3.7.

If an object overlaps any other object in the scene — caused by other individuals or the movement of boards or projector screens — then a single bounding rectangle (as shown in Figure 3.7d) will be created using this clustering approach. Such cases are not expected to have a substantive impact since the motion typically has a very short duration, and the presenter box will return to the correct size when the motion ends. However, shadows, lighting artefacts or student heads close to the presenter are clustered together, resulting in an oversized bounding box as shown in Figure 3.7f.



(a) Unmerged **(b)** Merged

(c) Unmerged **(d)** Merged

(e) Unmerged **(f)** Merged

Figure 3.7.: Bounding boxes are clustered/merged into a single box.

Clustering threshold A bounding box's *size* can be used to evaluate its validity. The *width* is less reliable since the presenter could be gesturing as seen in Figure 3.10c or walking. However,

the height of the box typically represents a close approximation of the presenters' height. Some exceptions occur when the presenter reaches up, is partially occluded (e.g. behind the podium) or due to poor differencing output.

Using a fixed height value threshold (in pixels) would be unreliable since the pixel density of different cameras would affect the threshold. The camera distance from the presenter, lens configuration or angle could also affect the threshold. However, the camera is typically mounted such that the field of view encloses only the front presenter stage area, and this area covers limited scene depth. Consequently, the presenter is relatively similar in size across a range of venue configurations. Using this assumption, we defined the following equations for approximating the presenter height with respect to the frame height. Figure 3.8 shows this relationship between presenter height (h_p), frame height (h_f) and occluded presenter height (h_{po}). The black regions below and above the image are due to a mask set on the cameras to exclude regions that were not of interest. These, however, do not affect the calculation. Recalibration of these parameters would likely be required when used in venues with different camera fields of view, aspect ratios or mounting configurations.

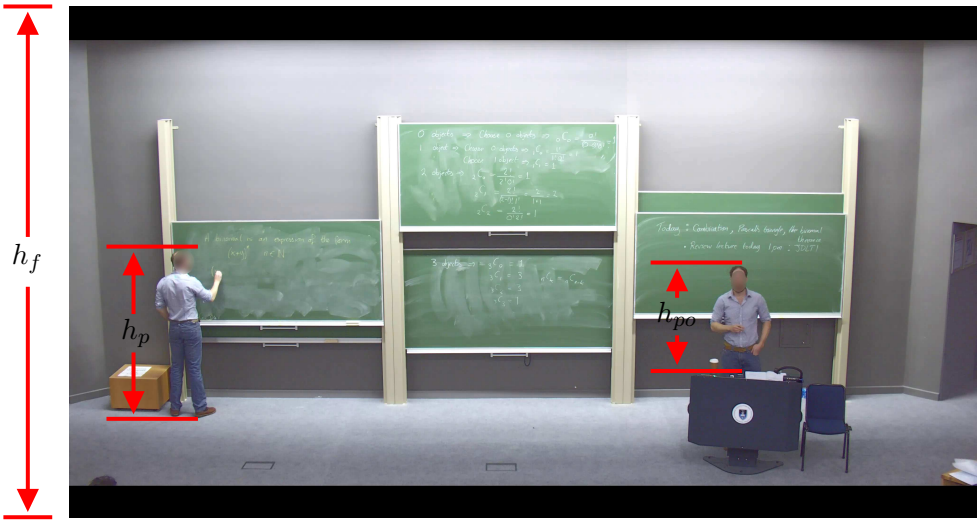


Figure 3.8.: Relationship between presenter height (h_p) and frame height (h_f) as well as an example of an occluded presenter height (h_{po}).

We define the average presenter height as a factor of the frame height as follows:

$$h_p = c \times h_f$$

The constant value c represents the *heightFactor* parameter and was determined to range between 0.1 and 0.5 with an average of 0.3 from a set of videos by calculating the average as follows:

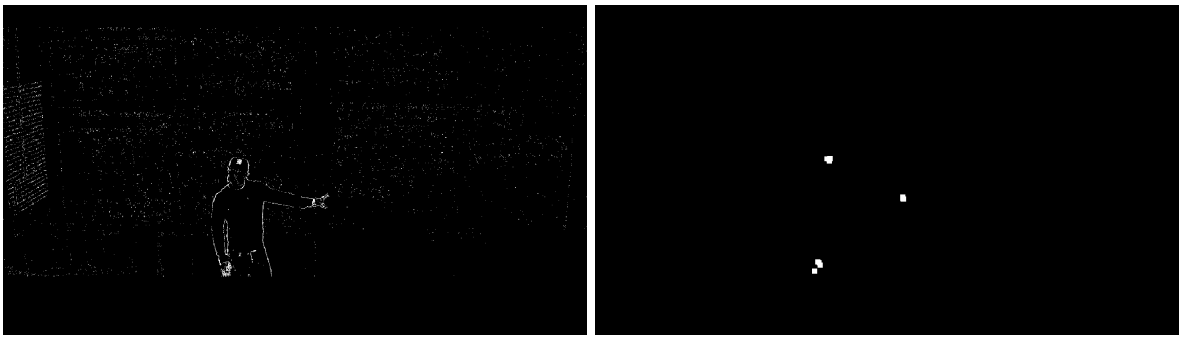
$$c = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{h_p}{h_f}\right)_i$$

where n is the set of training videos discussed in Chapter 4. However, to account for occlusion, it was visually observed that under typical lecture capture conditions, the presenter's height would very rarely be occluded by more than 50%. The final equation, accounting for occlusion, is thus:

$$\begin{aligned} h_p &= \frac{c \times h_f}{2} \\ &= \frac{0.3 \times h_f}{2} \end{aligned} \tag{3.2}$$

The h_p value is used as the clustering threshold and for ignoring small objects in gesture detection.

Limitations When the presenter is stationary, differencing results in very small values in the difference image, which are then eroded away by the morphology operations as seen in Figure 3.9. The filter heuristics discussed later in this chapter attempt to address this limitation.



(a) Before morphology **(b)** After morphology

Figure 3.9.: Figure A shows the output image produced by the threshold function of a presenter that is stationary, which produces a very thin outline. Figure B shows the same image after the morphology operations.

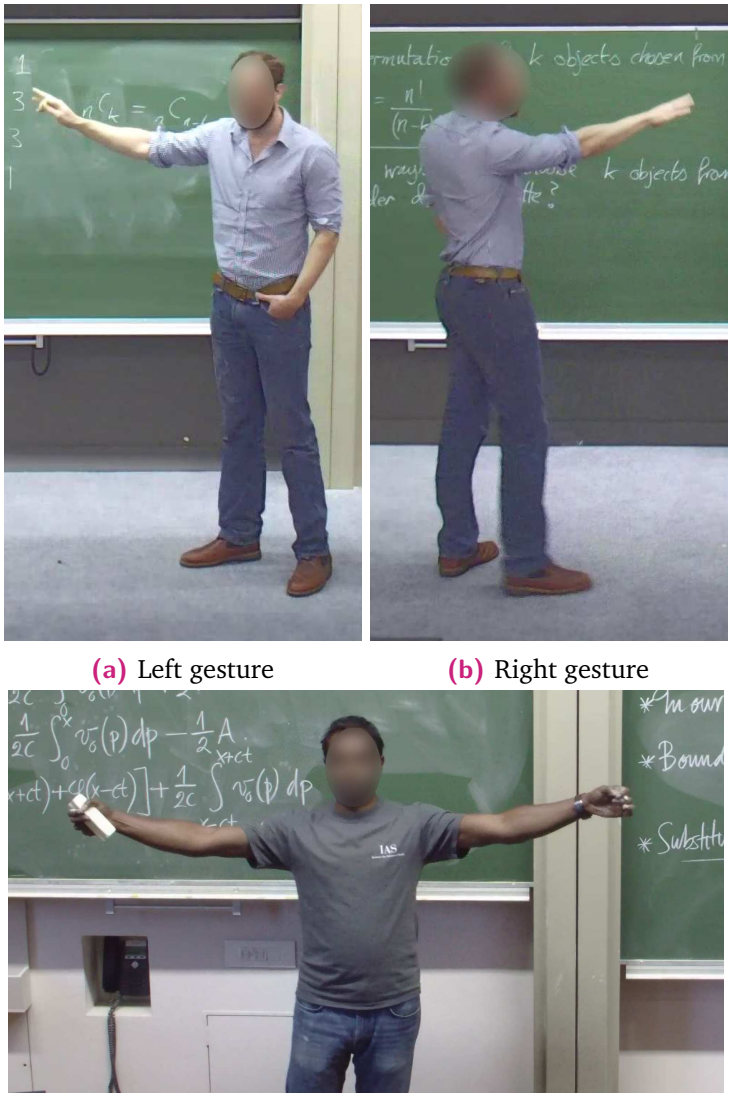
3.5 Gesture Detection

Design The potential of detecting gestures is investigated as an extension to the selected presenter detection method. In the literature review, the future work proposed by Wulff and Fecke [95] suggested using a set of heuristics that rely on the spatial distribution of pixels as a method of predicting gestures. In this section we follow a similar approach, by using image differencing (from the presenter detection stage) and the distance from the outermost pixels to the object centre, to predict large pointing gestures. This heuristic relies on the assumption that a gesture causes the majority of pixels (presenter body) to be skewed to one side of the frame. This imbalance can then be detected and used to indicate potential gestures. This method will only indicate a left or right direction, not an angle. However, combined with the board usage history, this information could potentially enhance a VC's framing decisions of recently used boards.

The type of gestures that we consider are "large and obvious" pointing gestures, with extended arms as shown in Figure 3.10a and 3.10b, rather than small obscure gestures. Furthermore, gestures similar to that shown in Figure 3.10c will not be detected using this algorithm due to the symmetry. However, these are not meaningful gestures for our application since there is no single direction.

To detect gestures, the known location of the presenter is required. However, at this pipeline stage, there is no clear way of knowing which *MovingEntity* (if any) is truly the presenter. The cropped difference image stored in each *MovingEntity* object, as discussed earlier, is now analysed to detect potential gestures.

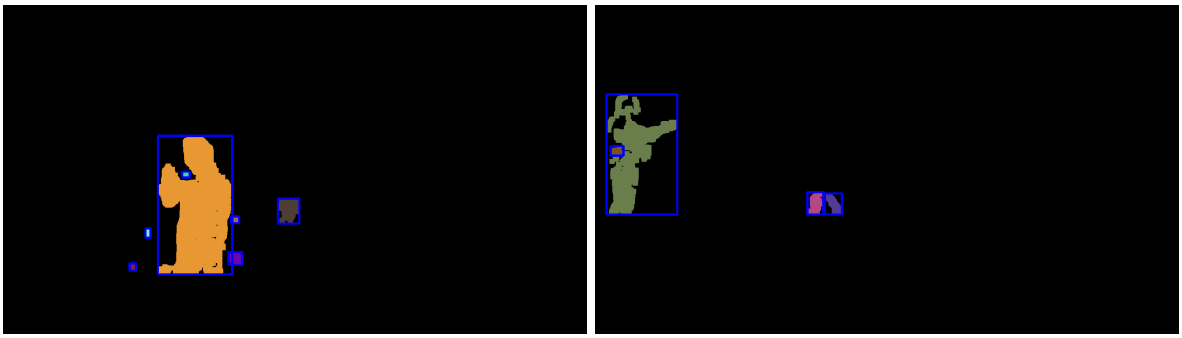
Ignoring small and oversized objects As seen in Figure 3.6 and Figure 3.11, the scene only contains one presenter and the other boxes are false positives and should be labelled as such. The *DataFilter* module is discussed later in this chapter and is executed after the video processing loop of the program to remove outlying data. Because the *GestureDetection* module is located in the loop (see Figure 3.1), it does not have access to the filtered data produced by the *DataFilter* module. To prevent searching obvious false positive *MovingEntity* objects for gestures, small *MovingEntity* objects are ignored by this module if their height is $< h_p$ using a *heightFactor* of 0.3 as defined in Equation 3.2. Similarly, *MovingEntity* objects with a height $> h_p$ using the upper bound *heightFactor* of 0.5 are also ignored as being oversized. Oversized *MovingEntity* objects are not as common and are generally caused by short-term events such as changing projector slides, moving the board as shown in Figure 3.7 or students entering and leaving the venue.



(a) Left gesture **(b)** Right gesture

(c) Double gesture (left and right simultaneously)

Figure 3.10.: Examples of left and right gestures seen in (a) and (b) as well as a double gesture seen in (c).



(a) Unwanted objects caused by shadows **(b)** Student heads in view

Figure 3.11.: Unwanted objects caused by shadows or students' heads in the view can be removed based on size.

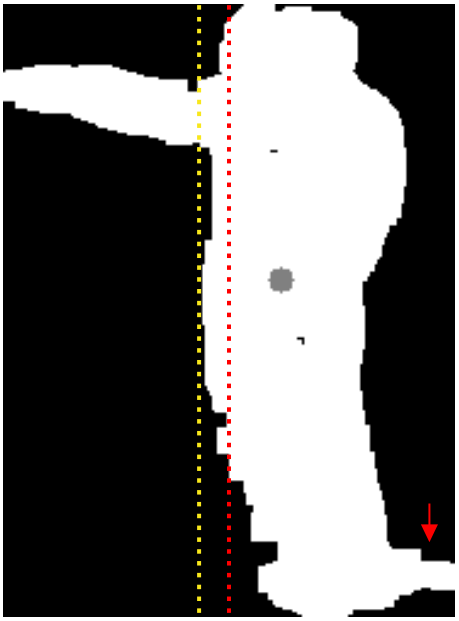


Figure 3.12.: Presenter bounding box partitioned in the centre (red line) and partitioned such that none of the body crosses the partition line. The red arrow indicates a shadow that is stretching the box, however, typically the centreline of the box would partition the arm from the body.

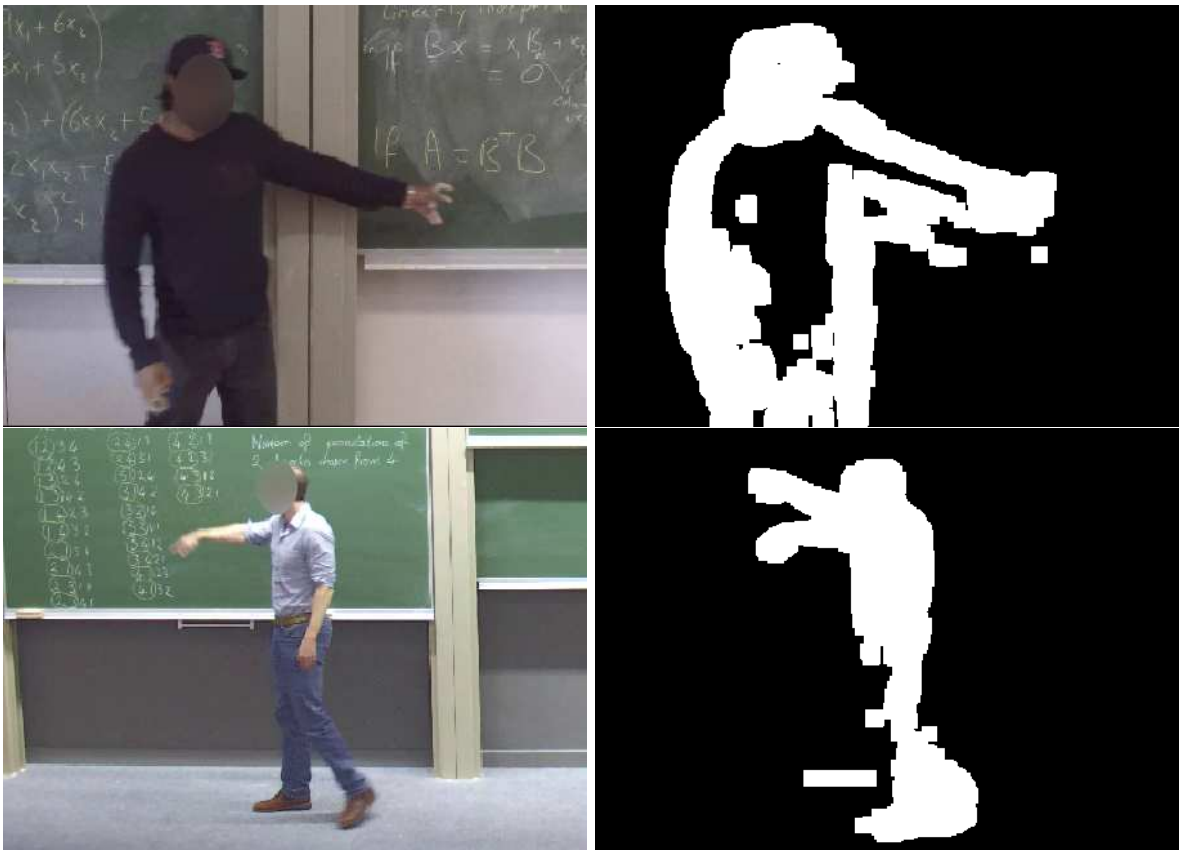


Figure 3.13.: Examples of malformed silhouettes that result in the proportional gesture detection method failing.

Search algorithm In a well-bounded presenter silhouette, a large pointing gesture typically shifts the presenter's body to one side of the bounding box, and by analysing the pixel distribution, gesture events can be predicted. Ideally, the bounding box is partitioned so that one side includes only the arm and the other side includes the body (as illustrated by the yellow line in Figure 3.12). However, calculating this partitioning line would require first determining the distribution of the pixels across the x-axis and then choosing an optimal partition point. If computational cost is not an issue, a robust step/edge detection algorithm could be used to analyse the graph produced, as shown in Figure 3.14. Instead, experimentation showed that an acceptable approximation is to simply partition the box at the midpoint (the red line in Figure 3.12), which requires considerably less calculation and is thus faster.

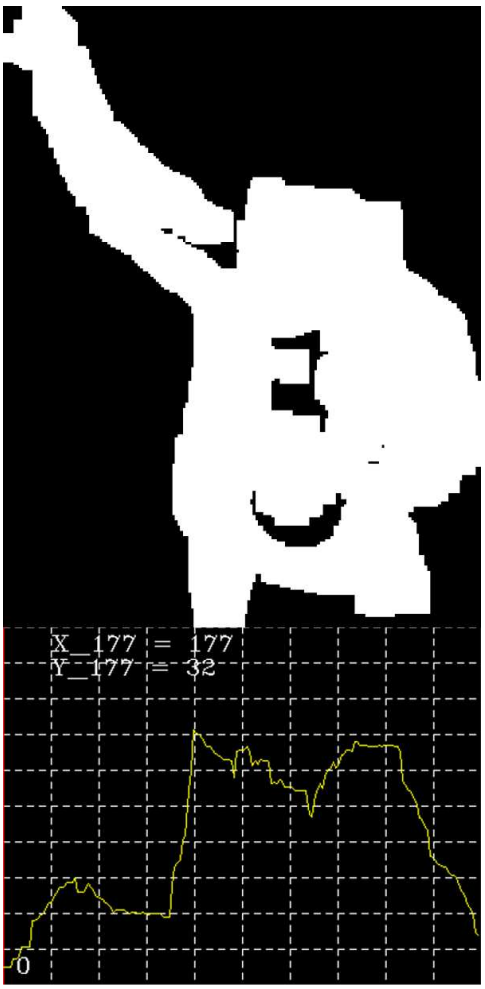
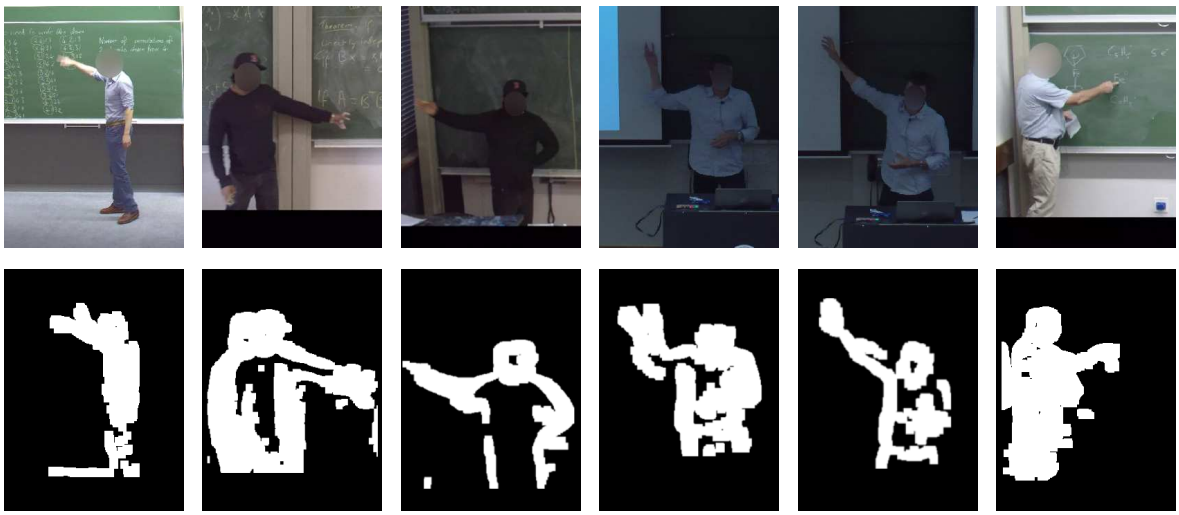


Figure 3.14.: The number of white pixels is summed per column and displayed in a line graph. A signal processing algorithm to detect the step could be used to calculate a specific partitioning line which separates the gesture from the body more accurately than the approximation approach of partitioning the box into halves.

Using this partitioning, the white pixels in each half are counted and expressed as a ratio of the total number of white pixels in the entire bounding box. By comparing the white pixel density differences between the two sides, a prediction can be made indicating a left/right gesture. To prevent this approach from detecting false positives caused by the presenter walking, as seen in Figure 3.15, the box is further partitioned into quarters Top Left (TL), Bottom Left (BL), Top Right (TR) and Bottom Right (BR). For each quadrant, the ratio of white pixels in that quadrant to all white pixels in the image is calculated. When a potential gesture is detected (i.e. the difference (Diff) between the left and right is greater than a set threshold), the top and bottom quadrants of that half are then compared to ensure that the top has more white pixels than the bottom. These ratio calculations also ensure that the approach is unaffected by variations in video resolution. Figure 3.16 shows a sample of pointing gestures and the pixel density by region, below each example.

Figure 3.15.: Example of a false positive gesture if only the halves are considered. Checking to ensure that the top quadrant has more white pixels than the bottom avoids this problem.

Limitations As previously discussed, the approach depends heavily on a well-defined silhouette and bounding box. Due to the range of unconstrained background colours and our chosen differencing approach, we were unable to achieve consistently well-defined silhouettes, as shown in Figure 3.12. Instead, many silhouettes were distorted, as shown in Figure 3.13. Furthermore,



(A)	(B)	(C)	(D)	(E)	(F)
Left = 0.227316	Left = 0.709704	Left = 0.336943	Left = 0.365979	Left = 0.330647	Left = 0.777681
Right = 0.772684	Right = 0.290296	Right = 0.663057	Right = 0.634021	Right = 0.669353	Right = 0.222319
Diff = 0.545367	Diff = 0.419408	Diff = 0.326114	Diff = 0.268043	Diff = 0.338707	Diff = 0.555361
TL = 0.161945	TL = 0.308942	TL = 0.297659	TL = 0.277152	TL = 0.266994	TL = 0.374186
BL = 0.0653709	BL = 0.400762	BL = 0.0392844	BL = 0.0888266	BL = 0.063653	BL = 0.403495
TR = 0.42003	TR = 0.157188	TR = 0.38212	TR = 0.253005	TR = 0.204504	TR = 0.156275
BR = 0.352654	BR = 0.133108	BR = 0.280936	BR = 0.381017	BR = 0.46485	BR = 0.0660445

Figure 3.16.: These images show example distributions of white pixels between the left and right half of the image and distributions per quadrant.

the clustering approach also produces oversized bounding boxes in some instances, as illustrated in Figure 3.7f, resulting in the failure of the algorithm since the proportions are no longer satisfied.

Preliminary testing shows that the current approach is plausible when supplied with well-defined silhouettes and bounding boxes. This approach is expected to work in environments with consistent backgrounds, such as a green screen. However, the proposed method did not produce reliable results due to the limitations of our differencing-based object detection approach. Lastly, a *skipFrames* value that is too large can also result in oversized bounding boxes or miss short-duration gestures entirely.

3.6 Writing Detection

Design Detecting regions on the blackboard surfaces or projector screens that the presenter uses aims to assist the VC in making better context-centred framing decisions by including these regions in the view.

Detecting board/screen usage regions is challenging since the approach needs to detect various media such as projector text, handwriting, images or drawings. These challenges are further exacerbated by visibility factors, such as lighting, contrast, size, and writing/text style.

Instead of developing a specialised detector (which is a separate field of research), we assume that writing or text is typically present at some stage during the presentation, and we explore the feasibility of using an existing text detector purely as a proxy for finding regions on the boards/screens that the presenter uses.

The literature review indicated that machine-learning-based text detectors are more robust under challenging conditions. Unfortunately, these approaches run significantly slower without using a GPU. However, unlike presenter detection, which requires a higher detection frequency, writing/text changes slowly over time and remains in the same location for longer. Thus, using a lower detection frequency to avoid adding much computational overhead enables us to explore a machine-learning-based approach without a GPU in our low-resource context.

Writing detection algorithm After having surveyed the literature, the EAST text detector proposed by Zhou et al. [106] along with their pre-trained neural network[1] was identified as a robust and state-of-the-art option to explore in our project as a means of detecting board/screen usage. To avoid the computational overhead of using this approach without a GPU, we introduce a *writingDetectionSkipTime* parameter that specifies a time delay (in seconds) between detections.

The Deep Neural Networks (DNN) module in OpenCV was used to call the pre-trained EAST network. The network (like many other DNN networks) expects a square image where the width and height are equal and both a multiple of 32. The global *workingDimension* parameter discussed in Section 3.4 limits the available image size in this module too. Although up-scaling a smaller image did produce slightly better results in some cases, it unnecessarily complicates the parameter choices. The *writingDetectionImgWidth* parameter defined in the *config.json* file is used to resize the image before passing it to the EAST network. The default value for this parameter is set to 0 and indicates that the image width will be calculated from the

[1] https://github.com/zxytim/EAST

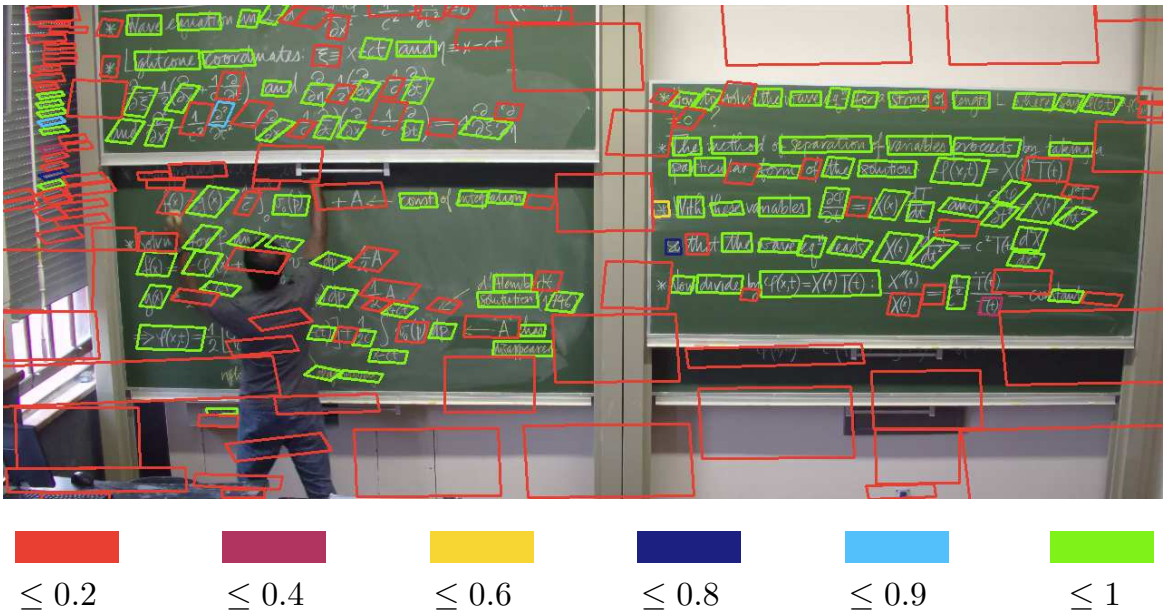


Figure 3.17.: Output produced by EAST and colour-coded confidence scores for the rectangles.

workingDimension width. If the *workingDimension* width is a multiple of 32 already, it is used as is and if not, the first value less than the *workingDimension* width which is a multiple of 32, is calculated and used. When the parameter is set to a non-zero value, that value is used as the width of the square image for EAST.

The decoded output from the network produces a list of rotated rectangles predicting text locations along with a confidence value for each rectangle. Figure 3.17 shows an example output from EAST with the colour of the rectangle coded to group confidence scores. In this figure, the rectangles appear sheared instead of rotated, however, this was due to an artefact of the drawing function used and are in fact, rotated rectangles as correctly illustrated by later figures in this section. The *writingConfThreshold* parameter is used to ignore all rectangles with a confidence score lower than this threshold. We found that the default confidence score of 0.5 set by the authors [2] worked well to eliminate false positive predictions. In Figure 3.17, a few boxes on the window blind have confidence scores between 0.61 and 1. While this threshold would not eliminate these, they are relatively uncommon occurrences and are not expected to have any substantive impact.

The EAST implementation also uses a Non-Maximum Suppression (NMS) algorithm to prevent overlapping predictions. Again, we found that using the default value of 0.4 for the *writingNmsThreshold* parameter worked well across our training set of videos.

[2] https://github.com/opencv/opencv/blob/master/samples/dnn/text_detection.cpp

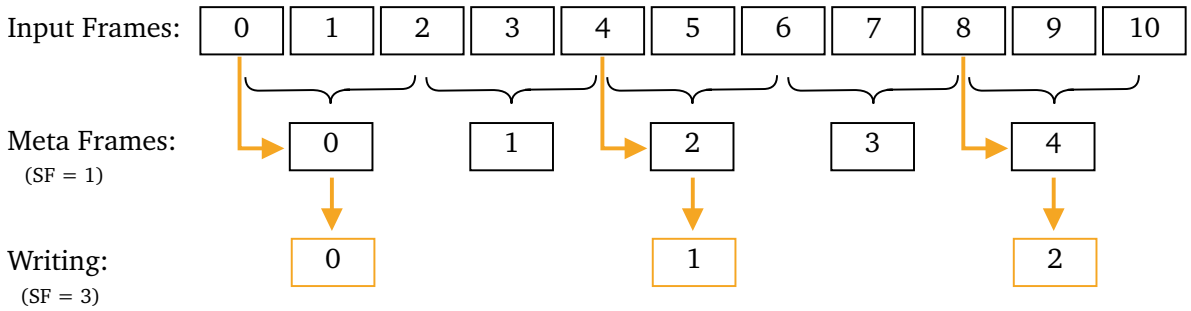


Figure 3.18.: This diagram illustrates the intervals between the *MetaFrame* and *WritingGroup* objects, and thus each *MetaFrame* does not necessarily contain writing information. The larger the interval between writing detection frames, the lower the resolution or accuracy of where in the input frames the text was changing. The entire range of input frames between two writing frames would have the same usage status determined by the two writing frames. The orange arrows indicate which of the two input frames that produce a *MetaFrame* are used by the writing detection module.

To reduce the frequency of calls to the EAST detector, an additional number of frames may be skipped in addition to the frames skipped by the *skipFrames* parameter. This skip parameter is called *writingDetectionSkipTime* and is expressed as a time (in seconds) in the *config.json* file. By multiplying this parameter value with the video frame rate, a number of frames to be skipped is derived and used only to detect writing periodically, as illustrated in Figure 3.18. A default time interval of 30 seconds was chosen through experimentation. Further analysis of these parameter values and their effects are demonstrated and discussed in Chapter 4.

Figure 3.19 shows four examples analysed by EAST with five of the commonly used input frame sizes. Each column is the same frame analysed with the dimensions shown in the first column. The first column also shows the average time (in seconds) to process a frame with that resolution. Column (a) and (b) shows a projector screen. The screen in column (a) contains very small text, only detected using the full resolution. This is a pathological case, and typical font sizes are similar to the example shown in column (b), which is well detected across all input resolutions. Columns (c) and (d) show an example of a green chalkboard, with column (c) having good chalk contrast and ambient lighting, while column (d) has a dusty board with lower lighting. A high chalk contrast and good ambient lighting produced better results in these preliminary experiments. Furthermore, a higher input resolution dramatically increases computation time and does not provide better detection results in most cases. Testing various input resolutions on a training sample of videos showed that an input resolution of 1280x720 yielded a good balance between runtime and accuracy for the videos. This is discussed further in the results chapter, Chapter 4.

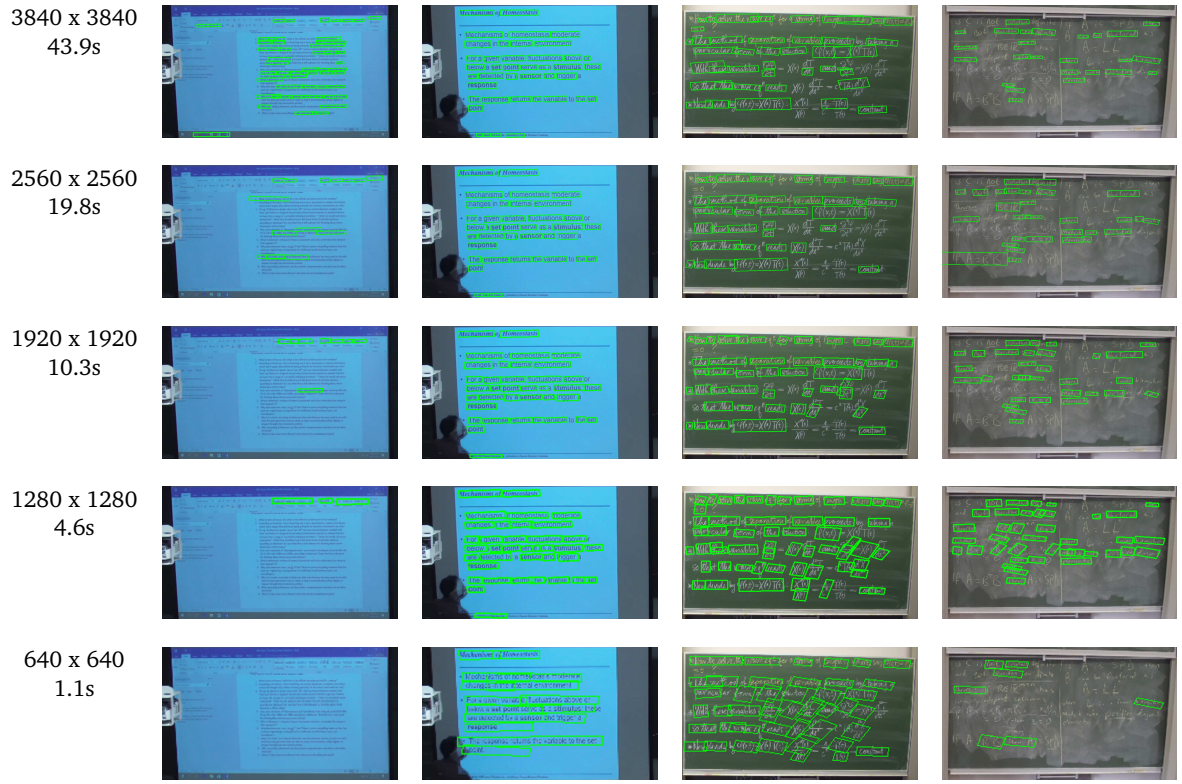


Figure 3.19.: Example output produced by the EAST network. The input resolution and average runtime (in seconds) are shown in the first column. Remaining columns (a) to (d) display four input images that are processed under commonly used input resolutions (rows). Column (a) and (b) shows text on a projector screen. The text in (a) is extremely small and is only detected well by EAST when using the full 3840-pixel frame width. Column (b) shows typical text size and is well detected across all input resolutions. Columns (c) and (d) show example boards with (c) having a clean board and good lighting and (d) being a dusty board under lower lighting conditions.

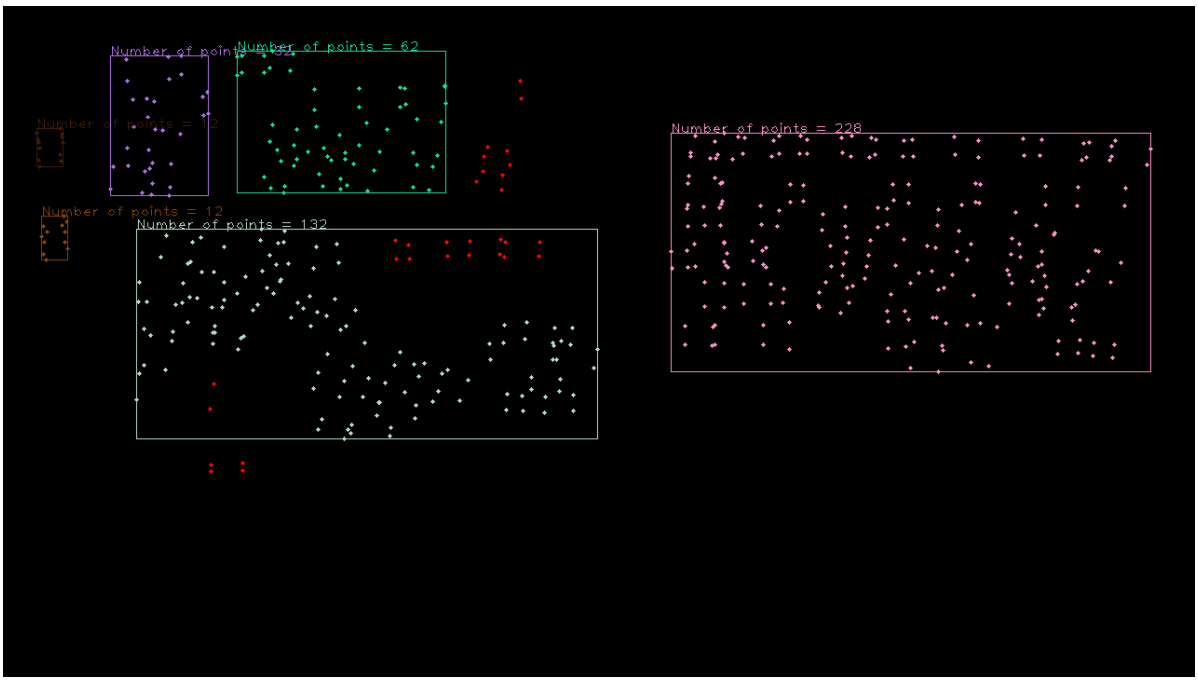


Figure 3.20.: An example of clustered vertices with a bounding rectangle around the clusters. Points in red do not belong to any clusters and are considered outliers (even though they happen to be included in the rectangle) since they could not be reached with the epsilon distance from any other points.

Clustering detected words The output produced by EAST consists of 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 to form regions of interest. This clustering also assists in including lost words (false negative predictions where legitimate words or text are given a very low confidence by EAST) that are excluded due to having a low confidence value, as seen in Figure 3.17. These cluster regions and the number of rotated rectangles that belong to each cluster would be the output of this module, indicating regions of potential text that may be relevant to the presentation.

The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) clustering method [27] was chosen to cluster the words since it does not require the number of clusters to be specified upfront. Instead, DBSCAN uses two parameters, a distance threshold epsilon ϵ and *writingClusterMinSize*. A random starting point is chosen, and any points within the ϵ distance become part of the cluster. The *writingClusterMinSize* parameter specifies how many points are needed to form a valid cluster; otherwise, that cluster is discarded as an outlier.

To spatially cluster the words, we need to associate a point (or points) with each word box (rotated rectangle). Initially, we considered using only the centroid of each box. However, this does not work well due to the variations in word lengths and because EAST often encloses

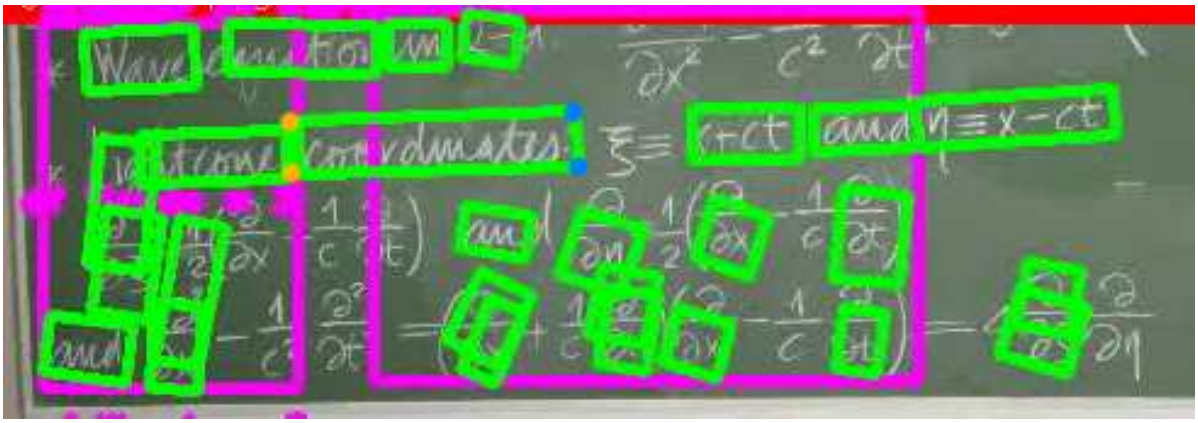
more than one word in a single bounding box. Instead, clustering based on a box feature, $f = [v_1, v_2, v_3, v_4]$ where v_i are the box corner vertices (as shown in Figure 3.20), produced better clustering results. The value for ϵ is set to the average "word" length which is calculated using the width of all the bounding boxes produced by EAST for that frame. Using the width keeps the parameter dynamic and adapts to any variations in font sizes throughout the video. Using the training dataset, we evaluated the effectiveness of using the average word length versus manually fine-tuning ϵ and the dynamic approach produced equivalent results (see Chapter 4, Section 4.5).

For the *writingClusterMinSize* parameter, a value of 12 was used to ensure that at least three words are needed to form a cluster. This eliminates individual words being clustered. These parameters can be changed in the *config.json* file but were chosen as the optimal default values through experimentation.

Clustering based on word box vertices often results in word boxes belonging to two separate clusters or not fully enclosed by the cluster boundary (magenta box in the figure), as shown in Figure 3.21a. To address these cases, the cluster boundary is expanded to include the entire word box when any of its vertices belong to that cluster as shown in Figure 3.21b. This correction can result in cluster bounding boxes overlapping and are resolved by merging them, as shown in Figure 3.21c. This merging process is recursive to ensure that newly merged clusters do not cause further overlaps, as illustrated in Figure 3.22. A recursive function responsible for adding the clusters to the vector was implemented to ensure that the cluster being inserted does not cause any overlaps. If it does, the overlap is merged, and the existing cluster is removed from the vector, and then the insert function is called with this newly created merged cluster. If no overlap is found, it is immediately appended to the vector, which is the function's base case.

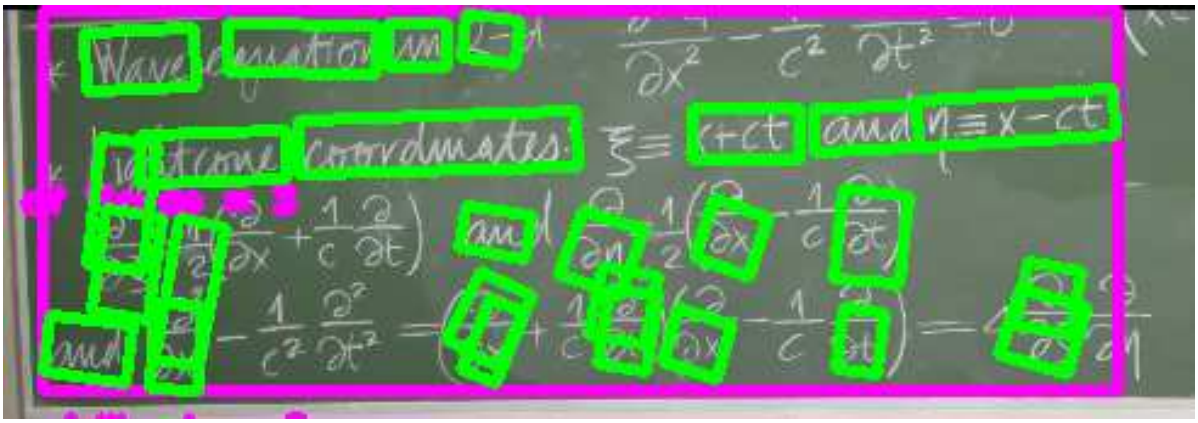
Limitations Partial occlusion of the text regions would result in the fragmentation of the clusters as illustrated in Figure 3.23 and Figure 3.24.

There are many instances where not all words are detected (as seen in Figure 3.17). This could be due to the model being too general and not explicitly trained in lecture contexts or similar environments. Furthermore, the pre-trained model does not detect diagrams, and images and performs worse on cursive handwriting styles. Figure 3.19 illustrates the degradation in performance for small fonts and dusty chalkboards.



(a) Clustering performed using the vertices of the word bounding boxes (green boxes). Each cluster is represented by a bounding box that encloses all vertices belonging to the cluster (magenta boxes). The long word with the orange and blue vertices belongs to two separate clusters.

(b) To resolve cases where a word belongs to two clusters or the word is not fully enclosed because the other vertices were out of reach, the bounding box is extended to include the whole word if any of its vertices are part of a cluster. As shown, this can then cause two clusters to overlap.



(c) Clusters that overlap, as shown in (b) above, are resolved by merging them into a single cluster, as shown here.

Figure 3.21.: Figures (a) to (c) illustrate how overlapping clusters are merged into a single cluster.

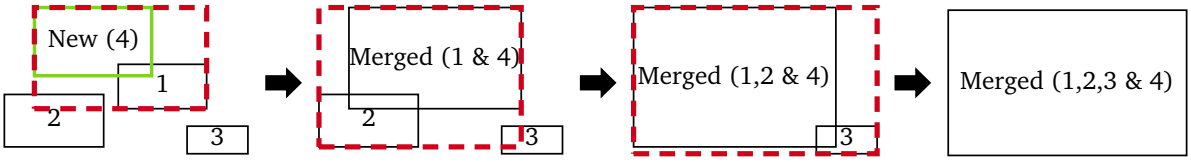


Figure 3.22.: This illustration shows the process for recursively merging clusters. In the starting image on the left, a new cluster rectangle (green) is to be added. This intersects with an existing cluster (1) and should be merged, as shown by the dashed red rectangle. This newly created merge now intersects with the existing rectangle (2). The process is repeated until all intersections are resolved and merged as shown in the final state on the right.

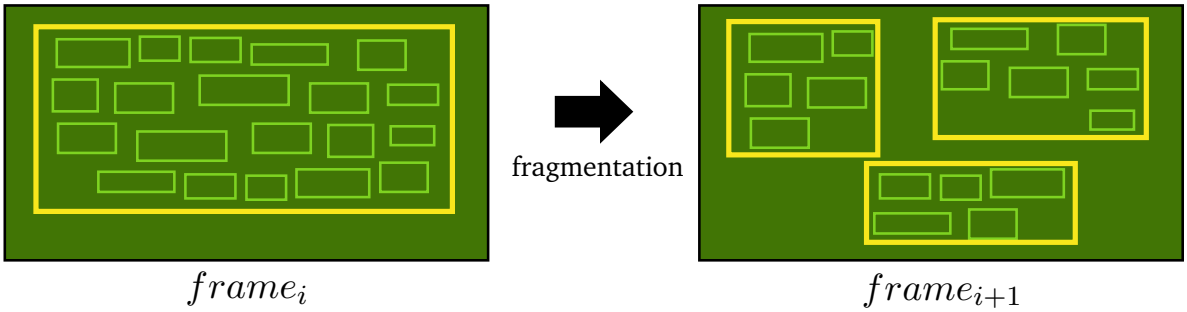


Figure 3.23.: Example of fragmentation which occurs when a single cluster fragments into multiple smaller clusters due to occlusion.

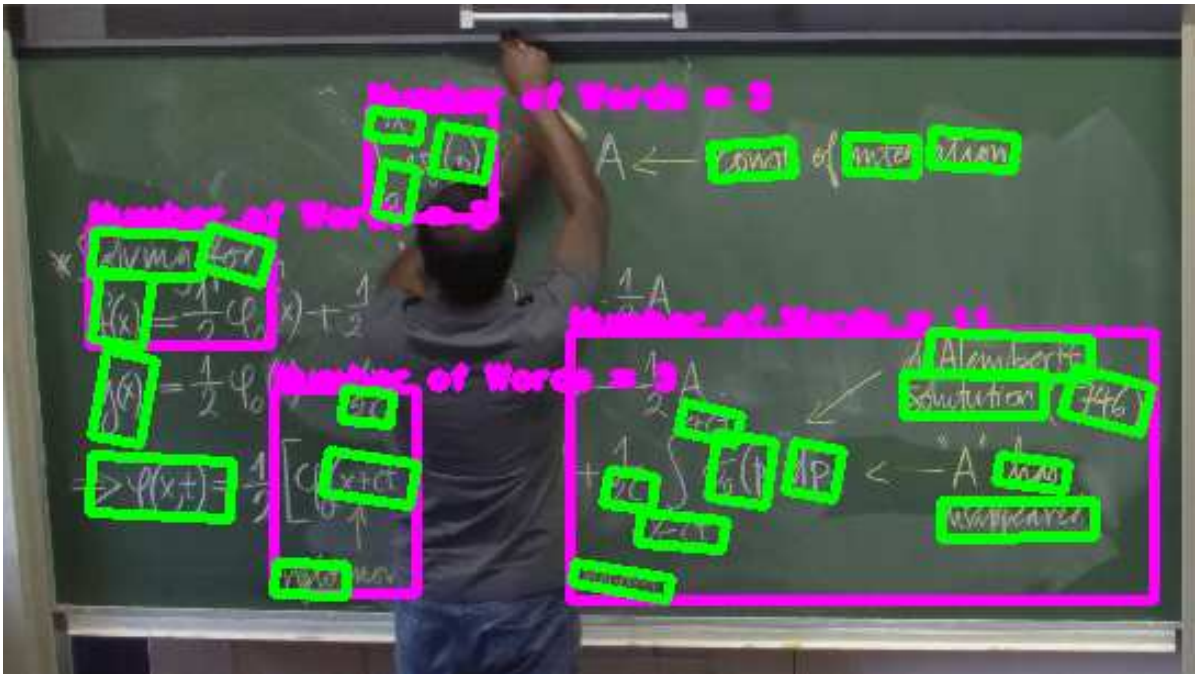


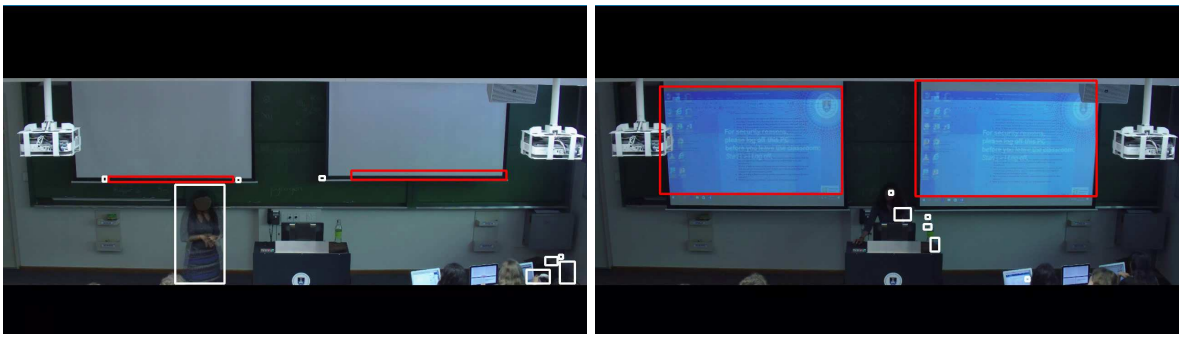
Figure 3.24.: Example of presenter occlusion causing fragmentation of the word clusters.

3.7 Data Filter Module

The *DataFilter* module is only executed after the main program loop and uses the mean and standard deviation of the detected objects' properties (discussed individually below) to serve as *adaptive* thresholds for eliminating unwanted bounding boxes in many of the following methods. The methods listed below are executed in this order.

Aspect and area ratio To remove abnormally large *MovingEntity* objects with unrealistic aspect ratios, we calculate the mean and standard deviation for the aspect and area ratio using all *MovingEntity* objects across all video frames. The aspect ratio is calculated as the $\frac{Width}{Height}$ of the *MovingEntity* and the area ratio is the *MovingEntity* area as a proportion of the total frame area. Any *MovingEntity* with an aspect or area ratio greater than one standard deviation above the mean is flagged as "deleted". They are only flagged and not deleted since the space has already been allocated, and erasing the items from the vector would take longer than setting a boolean status flag.

Figure 3.25 shows examples of bounding boxes with abnormal aspect and area ratios in red.



(a) Abnormal aspect ratios. **(b)** Abnormal area ratios.

Figure 3.25.: Examples of abnormal aspect ratio boxes (red) shown in (a) and abnormal area ratio shown in (b).

Vertical limits To remove remaining (unflagged) *MovingEntity* objects which are less likely to be a presenter, we calculate the mean and standard deviation for the vertical position (y-coordinate) of the object centroid across all frames. Using the standard deviation, we are able to create a band or region in which we expect to find the presenter object. Figure 3.26 shows an example with the amber lines representing one standard deviation above and below the mean (cyan line). Any unflagged objects whose centroid is not within one standard deviation from the mean (between the amber lines) are then flagged as deleted. This filter does not consider the object size and will remove all unflagged objects which do not meet this criterion. This band or

region is created because a presenter would normally walk horizontally across the stage area, and outlying objects would be excluded since they should not occur as frequently.

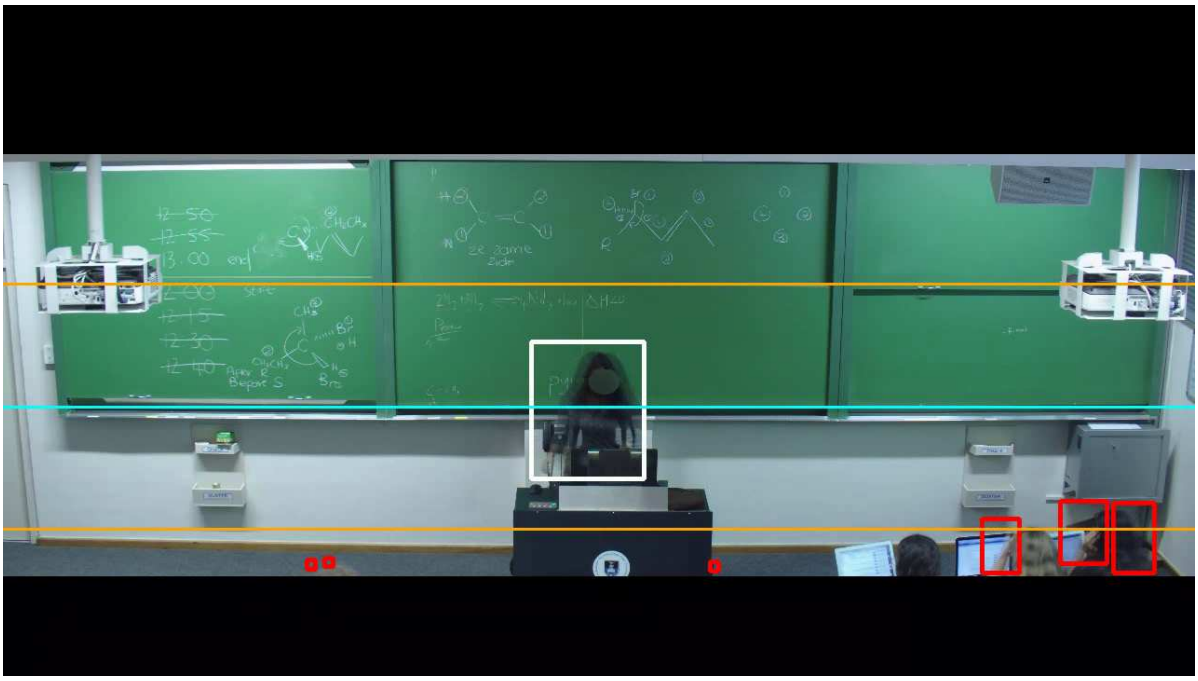
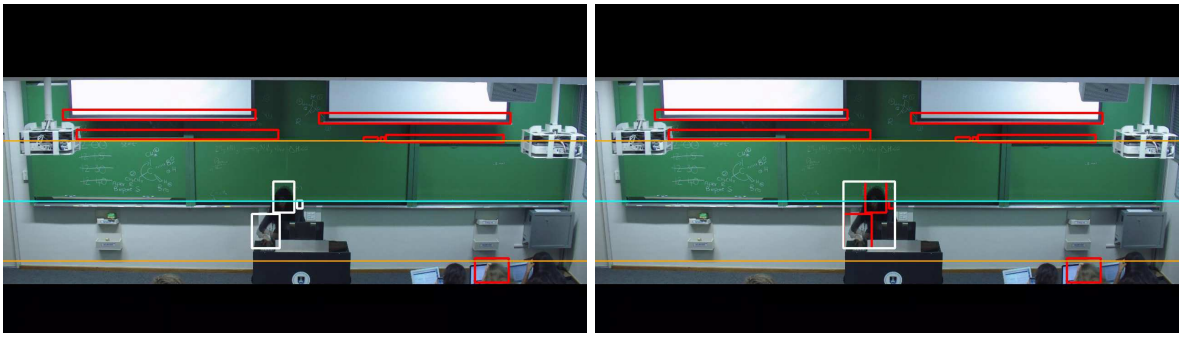


Figure 3.26.: The horizontal cyan line represents the mean vertical position of all box centroids over all frames, and the horizontal amber lines represent one standard deviation above and below the mean. Any boxes whose centroid is outside this region are removed (shown in red).



(a) Before clustering. **(b)** After clustering.

Figure 3.27.: Remaining white boxes in (a) are grouped due to their proximity to one another. A new *MovingEntity* object is created that encloses and replaces all old undersized objects as shown in (b).

Clustering *MovingEntity* objects across future and past frames When the presenter is relatively stationary, the presenter object becomes fragmented. After removing false positive objects using the previously discussed filters, we retrieve the unflagged *MovingEntity* objects from the previous frame and cluster them with those from the current frame being processed.

On the first frame, this does not apply since there are no previously processed frames. Using objects from the previous frame assists in frames with no motion (the presenter is completely stationary). However, using too many previous frames causes oversized clustered boxes, especially when the presenter moves around a lot.

For the clustering, we calculate the shortest distance between two *MovingEntity* objects vertices, and if the distance is less than a set threshold, the *MovingEntity* objects are merged, forming a single object with a new bounding box that encloses both objects. The mean height of all bounding boxes across all frames is used as the distance threshold. This process is illustrated in Figure 3.27.

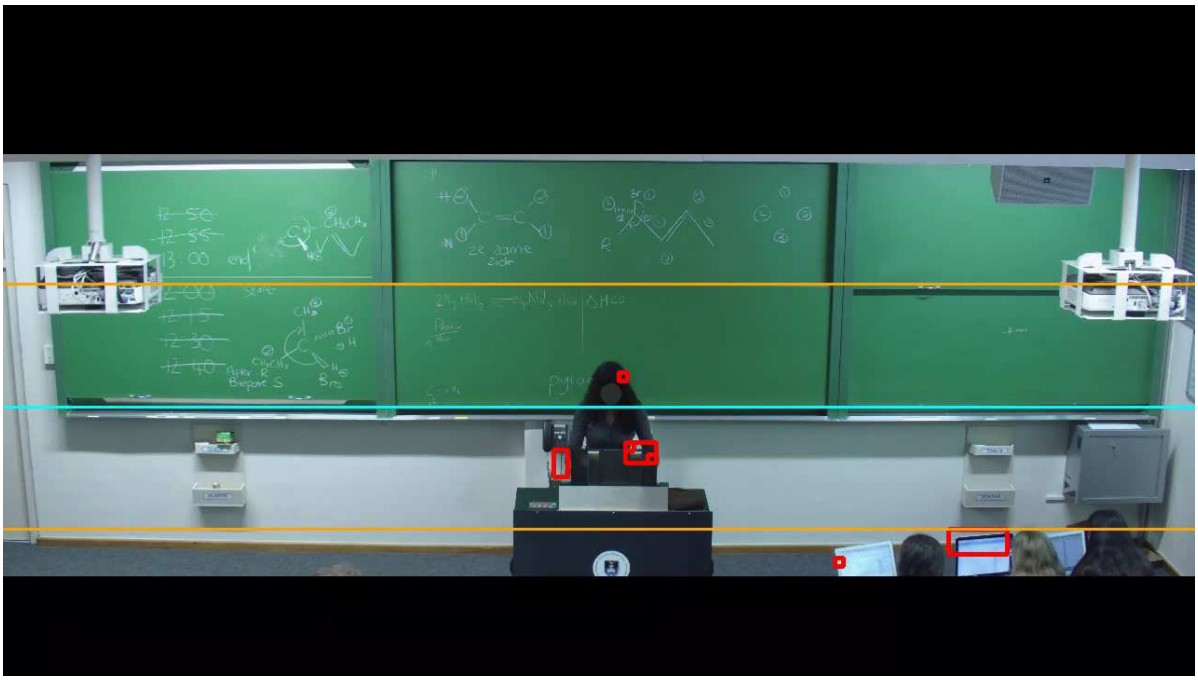


Figure 3.28.: The boxes around the presenter are flagged as undersized after clustering. There was insufficient motion, and clustering was unable to reach all boxes.

Interpolating *MovingEntity* objects After all the previous filters have been executed, we expect to have only one undeleted *MovingEntity* per person per frame. However, as shown in Figure 3.28, it is possible to have no valid *MovingEntity*. This method attempts to bridge the gap of frames where there is no valid *MovingEntity* by substituting an interpolated object between the frame with the last known good (undeleted) object (A) and the frame where a valid object (B) reappears in future. The object box size is calculated as the average size of objects (A) and (B). We only attempt to interpolate the gap if the start and end of the gap have a single valid object on each frame and if the gap size (number of frames without a box) is below a threshold. The mean gap size is calculated for all frames and used as the threshold.

Furthermore, because these frames without any valid *MovingEntity* objects are typically caused by a stationary presenter, we want to ensure that the gap is also only filled if boxes (A) and (B) overlap (potentially indicating a stationary presenter).

3.8 Implementation Language and Libraries

Open Computer Vision (OpenCV) [3] is an open-source computer vision and machine learning library with many highly optimised algorithms [12]. It had most of the algorithms required to implement the framework and was selected for this project. The C++ language is widely used in developing efficient image-processing algorithms and was selected as the development language for this project. Furthermore, we utilised the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) clustering algorithm from the Mlpack [4] open source library and statistical functions from the Boost [5] library.

3.9 Summary

This chapter introduced and discussed our project's design considerations and implementation details. An efficient frame differencing algorithm was used with minimal post-processing algorithms and heuristics to achieve presenter detection. The EAST detector was used to detect words (writing). Words were clustered into bounding boxes using the DBSCAN algorithm. The following chapter presents our evaluation methodology for testing these methods and discusses the results.

[3] https://opencv.org/
[4] https://www.mlpack.org/
[5] https://www.boost.org/

4 Evaluation and Analysis

This chapter details the methods and metrics used to quantitatively analyse our presenter, gesture and writing detection implementation. These modules need to be evaluated for accuracy, efficiency and robustness across a wide range of lecture scenarios.

We have identified commonly used metrics from the literature appropriate for evaluating each module and answering the research questions posed in Chapter 1.

4.1 Ground Truth Dataset

Data source: A set of lecture videos that included a large number of challenging lecture venue configurations were obtained from a university using 4K cameras. The dataset consists of 25 lecture videos, from several different courses and venues, with an approximate length of 45 minutes each.

We evaluated our work against labelled Ground Truth (GT) data. The labelling process is time-consuming and would require significant human resources to annotate full 45-minute lecture videos (approximately 80000 frames each). Instead, approximately 2-minute segments were clipped from the full-length videos (see Table E.1 and Table E.2 in Appendix E) covering a diverse range of behaviours and environments that are used for evaluation. These clips were purposefully selected to provide a wide variety of typical lecture behaviour and challenging scenarios. Examples include variations in lighting conditions, number of presenters/persons, movement, writing on boards, projector usage and different venue layouts.

The final dataset contained 100 (each approximately 2-minute length) video clips that were manually annotated frame-by-frame to produce the ground truth data. Table E.2 in Appendix E shows the breakdown of annotated objects and attributes for each video file. The annotation process is discussed next, which includes a discussion of the annotated objects and their attributes.

Annotation: Each frame of the selected videos in the dataset was annotated using the Computer Vision Annotation Tool (CVAT) [1] to be used as a GT for the evaluation. Figure 4.1 shows a screenshot of the CVAT interface with a labelled mathematics video.

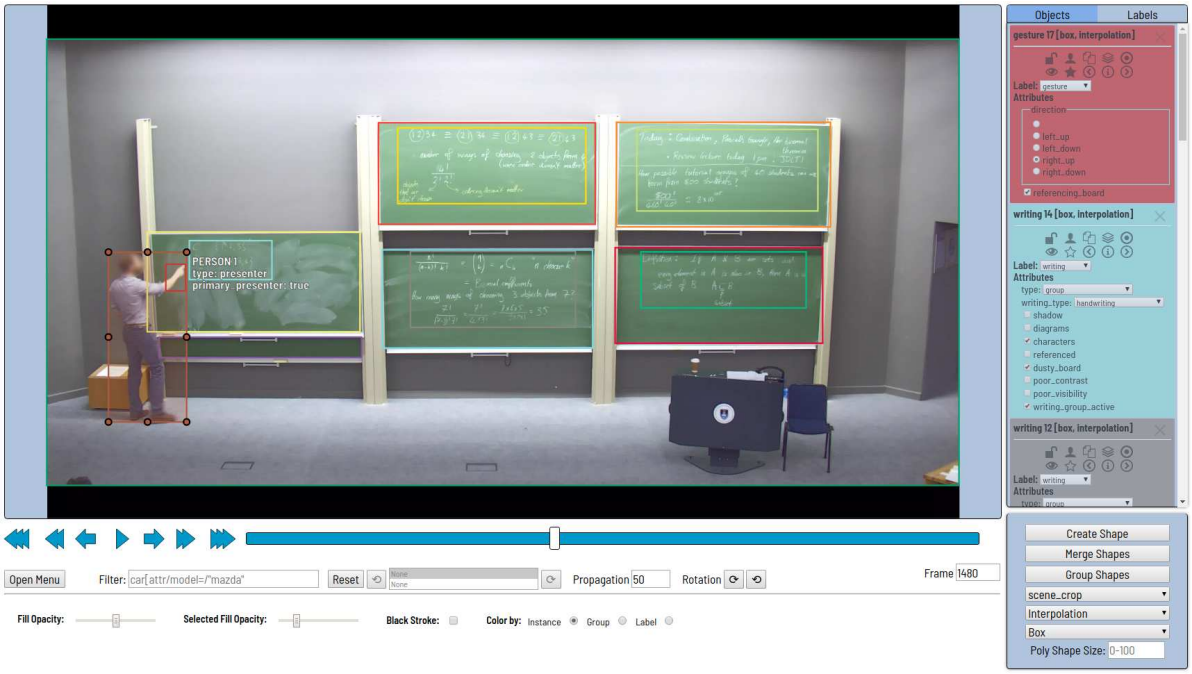


Figure 4.1.: Screenshot of the CVAT interface while labelling a mathematics lecture video.

This tool allows a user to manually annotate a video with a set of pre-configured labels, and all required objects were annotated with a bounding box as accurately as possible. The CVAT tool has an integrated TensorFlow object detection module that automatically annotates identified objects. This feature was used in the annotation process to annotate all "person" objects because there is a lot of motion generated by people (such as presenters and students). This motion results in a very time-consuming and costly manual annotation process. Furthermore, by using the auto annotation functionality, it standardises the labelling process of people objects resulting in a consistent output, leaving only objects such as gestures, boards, writing and lighting conditions to be manually annotated by a human annotator. The human annotator also verified and corrected any tracking issues made by the auto annotation and merged and labelled all bounding boxes of the known presenter to create persistence between all frames.

A set of annotation labels were defined to label objects (scene crop, person, gesture and writing) in the frames. Additionally, each label has a set of attributes that provide additional *environmental* insights for each label. These labels are detailed below with their attribute names indented and available options listed in [] brackets:

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

Scene crop Used to provide a cropping rectangle that excludes any black regions caused by privacy regions being set up on the camera. The attributes also provide overall scene information, such as being busy with many people in view and lighting conditions.

Lighting: [good, low] — See Figure 4.2 for examples of each of these lighting classifications.

Person Used to provide the location of all person objects in the scene.

Type: [presenter, student, interpreter, other] — Indicates the type of person. Only presenter and student were used as the other types were not present in our dataset.

Primary presenter: [true, false] — In a scene where there is more than one person object, this boolean indicates which one is currently presenting.

Gesture Used to provide the location of gestures.

Direction: [left-up, left-down, right-up, right-down] — Although there are four directions in the GT, we only require left or right for our evaluation.

Referencing board: [true, false] — Indicates a large pointing gesture if true and smaller obscure gestures are labelled as false.

Writing Used to enclose the text in the scene, which can be handwritten or on a projector screen. A single bounding box encloses all text on a single surface and is referred to as a writing group. Each board will have its own writing group (if writing is present). The attributes provide information regarding the board content. Visual examples of these attributes being used can be seen in Figure 4.3.

Type: [group, word] — The word label is only used if there is only a single word on the surface, otherwise, group is always used.

Writing group active: [true, false] — A writing group is marked as active the first time writing is added or removed and remains active until a new area becomes active.

Writing type: [text, handwriting]

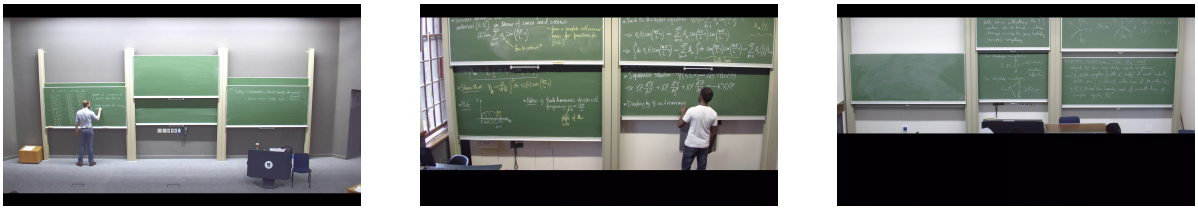
Poor visibility: [true, false] — A writing group is labelled with poor visibility when the content is not clearly readable.

Poor contrast: [true, false] — Poor contrast includes examples where the lighting is poor, the chalk colour is faint, or the board is very dusty.

Shadow: [true, false] — A word group is classified as having a shadow if a shadow is cast onto any of the enclosed text.

Characters: [true, false] — A writing group contains characters when there are standalone letters, symbols or mathematical equations/expressions in the writing group.

Diagrams: [true, false] — And writing groups that contain diagrams, drawings or images are flagged with this attribute. This does not provide a bounding box around the diagram, only that the writing group contains a diagram, image or drawing.



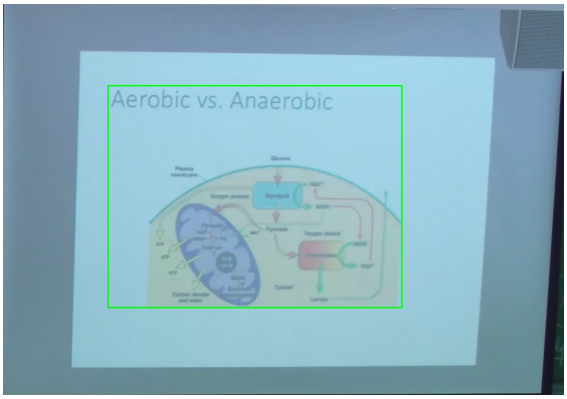
(a) Examples of scenes labelled as "good" lighting during annotation. These conditions are typically encountered when the presenter is using the writing boards, and the venue lights are turned on or blinds are opened for natural lighting.

(b) Examples of scenes labelled as "low" lighting during annotation. These scenarios typically occur more frequently when the projector is used and the venue lights are dimmed or turned off.

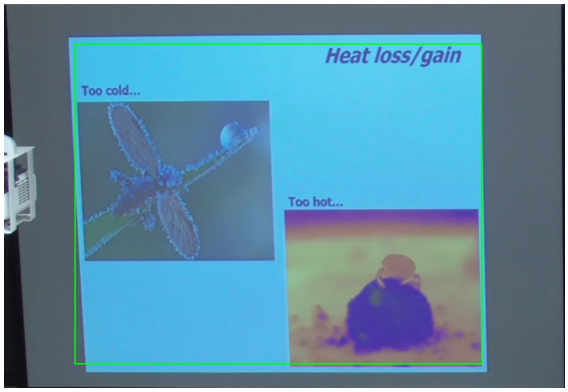
Figure 4.2.: These images illustrate the two classes of lighting conditions that were used when annotating the ground truth data. Good lighting conditions (a) are typically seen when the presenter uses the writing boards, while low lighting (b) is more prevalent with the use of the projectors.

Environmental factors: Some annotation labels (attributes) are used in the evaluation as environmental factors or conditions that could potentially influence the detection accuracy. For the presenter detection, we use the annotated *lighting* condition and three programmatically calculated factors: *multiple objects*, *student heads* and *presenter motion*. Frames with multiple ground truth objects are labelled as having *multiple objects*.

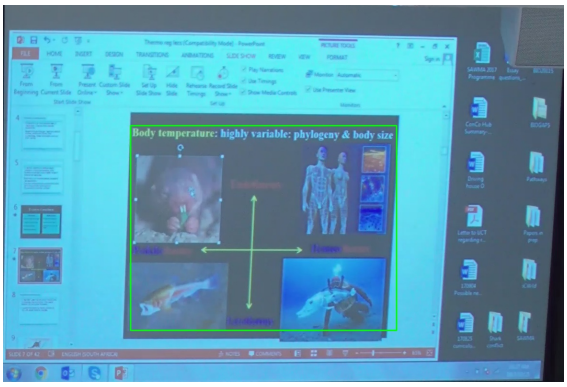
During the annotation process, students' heads were automatically annotated. An additional annotation rectangle was manually created to indicate the region where student heads were visible and were used to programmatically remove all the heads from the ground truth. This removal of student heads from the ground truth was necessary because the filter module in



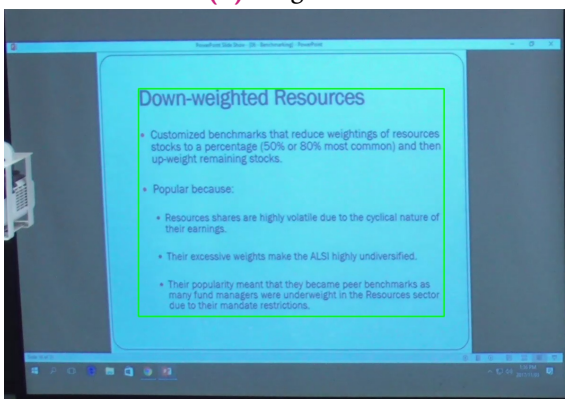
(a) poor visibility; diagrams



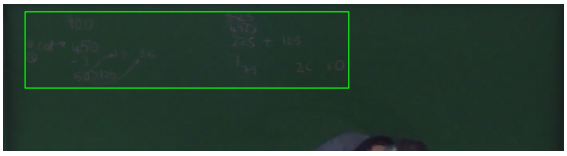
(b) diagrams



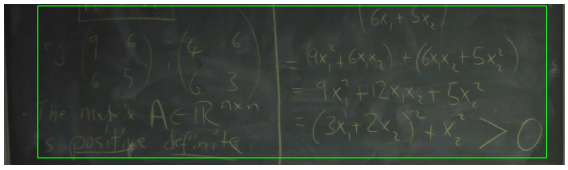
(c) poor visibility; poor contrast; diagrams



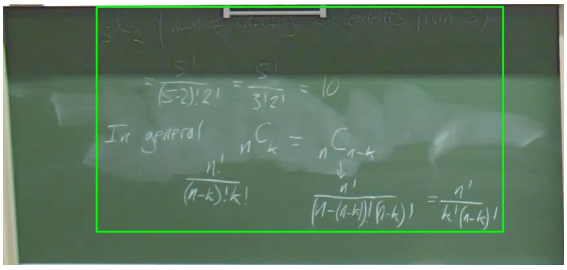
(d) no attributes



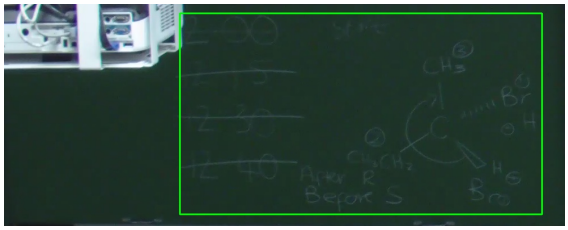
(e) poor visibility; poor contrast; characters



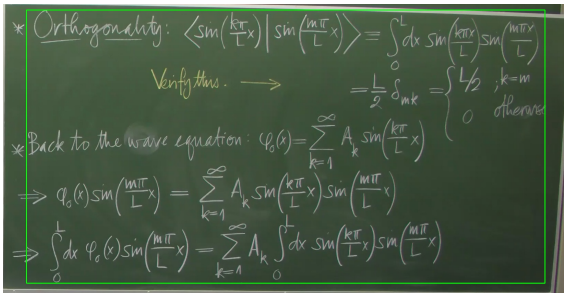
(f) poor visibility; poor contrast; characters



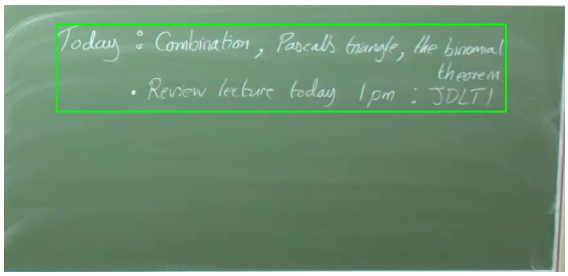
(g) poor visibility; shadow; characters



(h) poor visibility; poor contrast; characters; diagrams



(i) characters



(j) no attributes

Figure 4.3.: These images show examples of ground truth writing annotations with attributes used to describe the content.

our system aims to remove motion caused by students' heads. Thereafter, if our filter does not correctly remove these heads, it will decrease the overall *F1-Score* as expected. Since the presence of students' heads can affect our tracking, we keep track of frames where students' heads were present as an *environmental* factor.

Lastly, we calculate the *presenter motion* distance using the x coordinate of the presenter bounding box centroid between frames A and B (see Figure 4.4). This distance is then expressed as a proportion of the frame width to standardise the value across varying resolutions. These values are then split using the median to represent *low* and *high* motion categories with examples shown in Figure 4.7b.

The writing annotation labels are used as environmental factors that may influence the writing detection accuracy. The factors we consider are diagrams, lighting, projector text and visibility. Since there is typically more than one board on a frame, we sum up the count for each factor. For example, if any boards contain diagrams, the frame will be labelled as containing diagrams. For visibility, we group all conditions that affect visibility on the board (poor contrast, poor visibility and shadows) into a single factor, and when any of them are present, we consider the board visibility for the frames to be "reduced".

4.2 Training and Testing Datasets

The ground truth dataset is further partitioned into training ($n = 24$) and testing ($n = 76$) sets as shown in Table E.2 of Appendix E. We used an evolutionary optimisation solver in Microsoft Excel [2] to partition the dataset such that both sets have a similar distribution of objects and attributes (see Table E.3 of Appendix E). 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 these parameters have been calibrated. We reserved the smaller dataset to be used as the training set and the larger dataset for testing and analysing the environmental effects. Unlike typical machine learning applications, our implementation only has a few parameters to calibrate, and a smaller training set was sufficient. The larger set was reserved for the overall evaluation of the calibrated system and analysing the effect of the environmental factors.

[2] https://www.solver.com/analytic-solver-platform

4.3 Hardware and Runtime

The evaluation was performed on an *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)*.

We accumulate the runtime for each module in the system and export the data for evaluation to compare the accuracy of the object detection along with the runtime. We timed the execution of function calls using the *std::chrono::steady_clock* class in C++ which is recommended as being the most suitable clock for measuring intervals [3]. Below are the names of each timer and a short description of its usage.

elapsedTotal: Times everything from the beginning of the main function to just before the JSON file write method, because it must write this value into the file.

videoRead The time taken by all calls to the video read function includes reading, decoding and resizing the frame and storing it in memory.

motionDetection Times the call to the motion detection function, which performs the colour to grayscale conversion, frame differencing, morphological operations, thresholding, connected components analysis and creating *MovingEntity* and *MetaFrame* objects to hold the extracted motion information.

writingDetection Times the call to the writing detection function, which includes the resizing operation, blob conversion, network call, non-maximum suppression procedure and storing the detected word boxes to the *MetaFrame* object.

dataFilter Times the call to the data filter function.

4.4 Evaluation Metrics

Testing framework and statistical analysis We developed a python script to convert the annotated data into the same output JSON format produced by our software (see the JSON format used in Appendix D). This makes comparing the outputs of the annotations and our system simpler and easier to process.

A testing framework was developed in python and works by reading in the list of video file names in the dataset. Thereafter, the script executes our system with the filename and a set of parameters as input. Once execution is complete, the results of our system will be saved in a

[3] https://devdocs.io/cpp/chrono/steady_clock

JSON file. The testing framework then compares two JSON files. These two files contain the GT annotation data and the data produced by our system. The files are then passed into a set of test functions that evaluate each component (presenter, gesture and writing detection) with all of the identified metrics that will be discussed in the following section. Once each test metric is complete, the data from it is written into a file to be used for the statistical analysis.

All statistical analyses were performed using RStudio (version 2021.09.0 Build 351) with R (version 4.1.2).

Presenter detection Our presenter detection implementation does not "know" which objects are actual presenters and the remaining valid *MovingEntity* objects after performing the data clean-up as discussed in Chapter 3, Section 3.7, are all considered as "presenters". All the *MovingEntity* objects in each *MetaFrame* are compared to the equivalent GT frames. 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 4.4.

We use the popular Intersection over Union (IOU) method [68] to compare the system predictions to the GT for each *MetaFrame*. We have adapted and integrated their evaluation code [4] into our evaluation implementation.

The IOU measure aims to summarise the extent to which the predicted annotation (our system) agrees with the ground truth annotation. Equation 4.1 provides the definition of how the IOU is derived from the visual illustration in Figure 4.5.

$$\text{IOU} = \frac{Area(B_{GT} \cap B_S)}{Area(B_{GT} \cup B_S)} = \left(\frac{TP}{TP + FN + FP}\right) \tag{4.1}$$

Using the following classification criteria, the IOU is then used to classify the predictions as True Positive (TP) or False Positive (FP). The False Negative (FN) was not calculated explicitly since the way it and TP are used in the recall equation (See Equation 4.4) can be substituted with the GT count.

$$\text{Classification} = \begin{cases} TP & \text{if IOU} \geq 0.3 \\ FP & \text{if IOU} < 0.3 \end{cases} \tag{4.2}$$

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

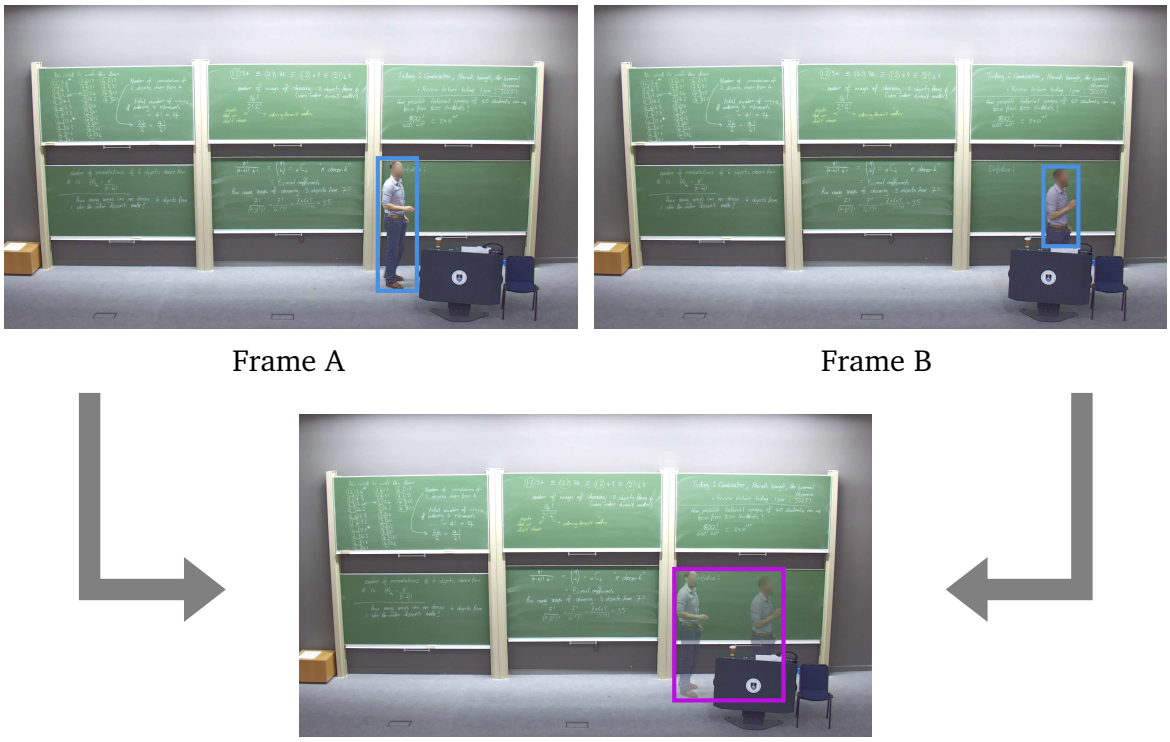


Figure 4.4.: The system *MetaFrames* are the product of differencing between two frames A and B (separated by the *skipFrames* value). To evaluate the detected objects in the system *MetaFrame*, they are compared to a ground truth equivalent *MetaFrame*. All ground truth frames were annotated, and to generate an equivalent *MetaFrame*, we take the union of the individual bounding boxes (cyan) between ground truth frames A and B to produce the merged region shown in purple. This purple region becomes the ground truth presenter for the particular *MetaFrame* being evaluated.

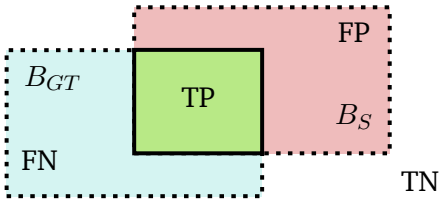


Figure 4.5.: An illustration showing the overlap between the ground truth bounding box B_{GT} and the system bounding box B_S.

A minimum IOU threshold of 0.5 is typically used [68], however, in our project, we do not require a precise overlap but rather an approximate location of the object. Using an IOU of 0.3 achieves this goal for evaluation, and Figure 4.7a shows two examples of an $IOU = 0.3$. An external virtual cinematographer should be capable of framing the best view despite not having the precise location of the presenter. 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.

After the system predictions have been classified, the precision, recall and *F1-Score* are calculated for each *MetaFrame* and provide an overview of the detection. The precision, recall and *F1-Scores* are defined as follows:

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{\text{Number of TP detections}}{\text{All system detections}} \tag{4.3}$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{\text{Number of TP detections}}{\text{All ground truth detections}} \tag{4.4}$$

$$\textit{F1-Score} = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4.5}$$

Gesture detection Gesture data is attached to the *MovingEntity* object in both the GT and S data. We use the IOU to establish if there is an overlap between the S and GT *MovingEntity* objects. If there are multiple overlaps, the one containing gesture information is used. In this case, we consider an IOU > 0 as a valid overlap, and the gesture information (if present) can be evaluated.

The GT gestures were annotated for each frame using the same videos used for the presenter evaluation. Because the implementation used for detecting gestures works better with large pointing gestures, we annotated such gestures as "referencing" in the ground truth when they were referencing a board or something else. This will provide insight into the implementations' performance for both types of gestures.

A frame-by-frame accuracy of when the gesture begins and ends is not essential. However, it is essential to know that there is a gesture and its direction. The gesture direction could be calculated as the angle of the forearm, however, this is not a simple problem to solve because presenters would often refer to a board or a region in a waving motion, making it challenging to know the exact angle to choose. Instead, knowing if the gesture is pointing to the left or right

of the screen was a more straightforward problem and would also provide useful information for making framing decisions when paired with the writing detection data.

Our system attaches gesture information to a parent *MovingEntity* — potentially the presenter — since a "gesture" is reasonably assumed to be an action caused by a (tracked) person. The gesture information includes the direction of the gesture stored as a vector and a boolean value to indicate left or right for faster comparison. The GT data includes the same data as the S but includes an additional boolean value that indicates whether the GT gesture is a "referencing" gesture or not.

Because presenters each have different styles of gesturing, we annotated all gestures in the GT, even the less obvious ones, but labelled the clearly pointing gestures as "referencing". This allows us to use this more detailed data in our analysis to understand our implementation's performance better.

Nickel and Stiefelhagen [64] use precision and recall along with an angle of error to evaluate the performance of their gesture detection implementation. Our implementation only affirms the presence and left or right direction of a gesture, and the angle of error metric does not apply to our evaluation.

Each gesture is classified using the following criteria:

$$\text{Classification} = \begin{cases} TP & \text{if IOU } > 0 \text{ \& GT direction} = \text{S direction} \\ FP & \text{if IOU } > 0 \text{ \& GT direction} \neq \text{S direction} \end{cases} \tag{4.6}$$

The IOU refers to an overlap between the GT and S *MovingEntity* objects since the gesture information is a child object of the *MovingEntity* object, and we only want to compare potential gestures that belong to the same object in the scene. We use an IOU of > 0 since we do not want a poor-performing object detection to affect the results of the gesture evaluation, so any overlap is sufficient to evaluate the gestures. These classifications are then used to calculate the precision (Equation 4.3) and recall (Equation 4.4) for the gesture detection on each *MetaFrame*.

Writing detection We followed the ICDAR 2011 competition evaluation protocol proposed by Wolf and Jolion [93] for evaluating our writing detection module since it accounts for fragmented detection (one-to-one, one-to-many and many-to-one). Instead of a bounding box

enclosing an individual word or sentence, we use it to enclose a block of text. We adapted an existing Python implementation [5] of these metrics to work in our evaluation framework.

Two overlap matrices are calculated that represent the area precision and area recall for all the combinations of GT and S detections. Finding matches between the GT and S rectangles is done by thresholding the values in these matrices and any element with a value greater indicates an overlap between GT and S. A quality constraint threshold $t_p \in [0, 1]$ is defined for the area precision and $t_r \in [0, 1]$ for area recall. A correct detection occurs when both these thresholds are satisfied.

The matches can be classified into three categories as discussed below:

- **one-to-one:** A single GT matches a single S detection.
- **one-to-many (splits):** A single GT matches many S detections.
- **many-to-one (merges):** A single S detection matches many GTs.

To accommodate these three types of matches, the precision and recall equations used for evaluating the writing detection are defined differently than those discussed in the previous section. The precision and recall equations are defined by Wolf and Jolion [93] as follows:

$$\begin{aligned} Precision(\mathbf{GT}, \mathbf{S}, t_r, t_p) &= \frac{\sum_i Match_S(\mathbf{GT}, \mathbf{S}, t_r, t_p)}{|\mathbf{S}|} \\ Recall(\mathbf{GT}, \mathbf{S}, t_r, t_p) &= \frac{\sum_j Match_{GT}(\mathbf{GT}, \mathbf{S}, t_r, t_p)}{|\mathbf{GT}|} \end{aligned} \tag{4.7}$$

where **GT** and **S** are vectors containing ground truth and system rectangles respectively for the frame and i and j are the indices in these respective vectors. $Match_{GT}(\mathbf{GT}, \mathbf{S}, t_r, t_p)$ and $Match_S(\mathbf{GT}, \mathbf{S}, t_r, t_p)$ are functions that return a value based on the type of match between GT and S and are defined as follows:

$$Match_S(\mathbf{GT}, \mathbf{S}, t_r, t_p) = \begin{cases} 1 & \text{if } S_i \text{ matches single GT rectangle} \\ 0 & \text{if } S_i \text{ does not match any GT rectangle} \\ f_{sc}(k) & \text{if } S_i \text{ matches multiple (k) GT rectangles} \end{cases} \tag{4.8}$$

[5] https://github.com/liuheng92/OCR_EVALUATION/blob/master/Algorithm_DetEva.py

$$Match_{GT}(\mathbf{GT}, \mathbf{S}, t_r, t_p) = \begin{cases} 1 & \text{if } GT_j \text{ matches single S rectangle} \\ 0 & \text{if } GT_j \text{ does not match any S rectangle} \\ f_{sc}(k) & \text{if } GT_j \text{ matches multiple (k) S rectangles} \end{cases} \tag{4.9}$$

where $f_{sc}(k)$ is a function used to calculate a penalty for splitting and merging detections. The authors used a value of 0.8 in their work. Although in our case, accuracy is not as important as getting sufficient overlap to identify when there is a board in a particular region, for comparison with their work, we set our penalty value to 0.8. The *F1-Score* is calculated using Equation 4.5. The default quality constraint thresholds defined by the authors (area precision threshold $t_p = 0.4$ and area recall threshold $t_r = 0.8$) are also used for plotting the performance diagrams shown in Figure 4.11.

4.5 Parameter Calibration

We use the training dataset to explore the effects of different system parameter combinations and find optimal values. These parameter values obtained in this calibration process are then set in the system configuration and used for final testing with the test dataset of videos. All system parameters have been set to reasonable defaults during development. In Section 4.4, the IOU threshold was used to classify detections as true positive or false positive and the value of 0.3 was used in all the parameter calibration experiments.

In this section, we keep all parameters fixed to their defaults and vary only one at a time to find the best parameter value experimentally. While this may not provide the global optimal parameter set, it avoids a combinatorial search of the parameter space and works well in practice. Exploring all parameter combinations would be impractical due to the long runtimes on our dataset.

After the best parameter value has been found, it is updated and the experiment is repeated for the next parameter. Figure 4.6 shows the parameter values considered for each parameter and the result thereof. This process is done for all parameters until all have been calibrated, by selecting the parameter value which yields the best *F1-Score* or runtime. The calibrated parameters are then fixed in the system configuration to be used in the final evaluation to determine the overall accuracy (using the metrics discussed in the previous sections) of the system using all videos in the test dataset as input.

A precision-recall curve is a very popular plot used to evaluate object detection implementations [68], however, our system does not produce confidence scores for detected objects, which is crucial for plotting precision-recall curves. Instead, we use the *F1-Score* as the primary metric in our parameter calibration.

4.5.1 Presenter Detection

As discussed in Chapter 3, the *skipFrames* and *workingDimension* can affect the object detection results. All the runtimes discussed in the following sections are with respect to the 2-minute video clips in our dataset.

The effect of the *skipFrames* value on runtime and *F1-Score* using a *workingDimension* of $720p$ is shown in Figure 4.6a. The *F1-Score* began to decline when *skipFrames* > 30 and *skipFrames* $= 28$ was selected as an optimal value resulting in the best *F1-Score* and runtime. Figure 4.6b shows the effect of downscaling the input 4K frame resolution on runtime and *F1-Score*. The video read time includes the decoding and downscaling operation times, 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 the runtime of the other processing modules. The majority of time is consumed by the video read module (disk IO and decoding operations).

Using a downscaled resolution of $720p$ provided the best *F1-Score* (0.8) for object detection and writing detection module shown in Figure 4.6c. The 4K resolution produced the lowest *F1-Score* (0.69). We selected $720p$ as the default resolution. The mean runtime for the motion detection module decreased from 13.7 seconds for the 4K resolution to 1.4 seconds for the 720p resolution $(89.9\%$ decrease). The overall runtime $(56.6$ seconds) using the 720p resolution suggests that the video can be processed in less than half the input video length. At a 720p resolution, the majority of the runtime (89%) was consumed by the video read module, which is unavoidable.

4.5.2 Writing Detection

The effect of the *workingDimension* on runtime and *F1-Score* are shown in Figure 4.6c. The $720p$ produced a better *F1-Score* with a mean runtime of 4.58 seconds as opposed to 1.12 seconds for the $360p$ resolution on a 2-minute clip. Figure 4.6d 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 detection frequency, and any variations in *F1-Score* are just due to different text

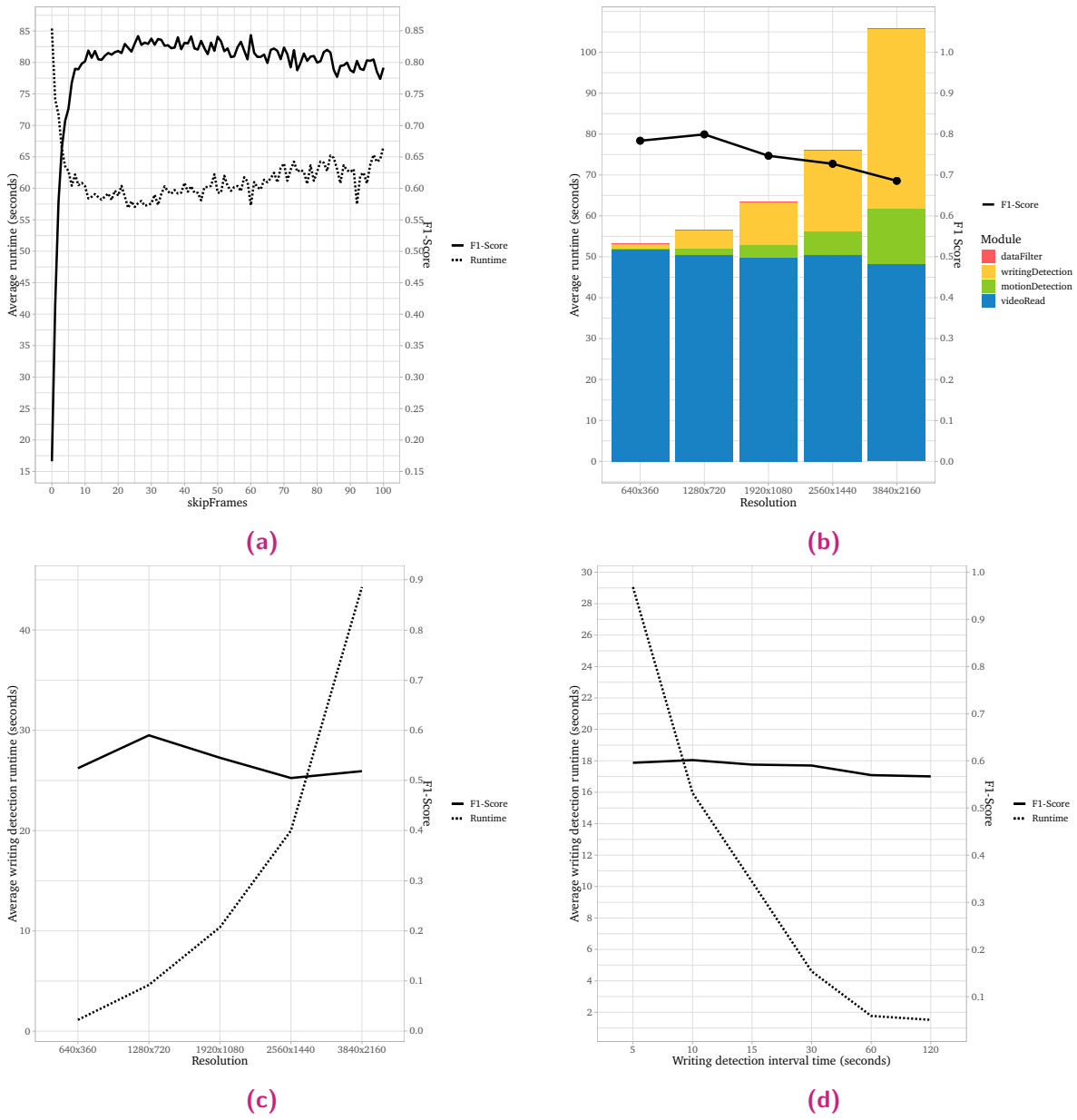


Figure 4.6.: Effects of (a) *skipFrames* vs. runtime and *F1-Score*. (b) *workingDimension* vs. module runtime and *F1-Score*. (c) *workingDimension* parameter vs. writing detection runtime and *F1-Score*. (d) *writingDetectionSkipTime* vs. writing detection runtime and *F1-Score*.

at those times. We also compared the outcome of the writing detection *F1-Score* when using the average word length as the clustering threshold discussed in Chapter 3, Section 3.6, as opposed to manually selecting the best threshold. The manual calibration only produces an extremely small improvement from 0.590 to 0.597, indicating that the chosen method works very well.

4.6 Results and Analysis

The results presented in this section are generated by running our system on the final, reserved set of data, previously introduced as the test dataset. At this stage, all system parameter values are fixed to the optimal defaults that were obtained through the calibration process described in the previous section. All videos are processed through our system and the output for each module (presenter, gesture and writing detection) is compared to the respective annotated ground truth equivalent using the respective evaluation metrics introduced in Section 4.4. All computation was performed using the hardware detailed in Section 4.3.

4.6.1 Presenter Detection

The presenter detection module is evaluated using the *F1-Score* obtained per *MetaFrame* and used to report the overall accuracy and model the impact of environmental factors introduced in Section 4.1.

Table 4.1 shows the frequency distribution of *F1-Scores* obtained for all the *MetaFrames* in our test dataset. From the table, we observe a multimodal distribution of *F1-Scores* with the majority skewed to 1. The high frequency of 1 is observed since the majority of both ground truth and system frames only have one object, as shown in Table 4.2, leading to the binary *F1-Score* of either 0 or 1 (the object was either detected correctly or incorrectly). The other *F1-Scores* are produced by the remaining combinations of ground truth and system object counts. In reality, an *F1-Score* $= 1$ is rarely obtained, however, it is important to remember this observation and to note that when evaluating the environmental factors, we are referring to an *F1-Score* that is calculated per frame and not an overall *F1-Score*. Our overall *F1-Score* is calculated once using all the frames.

Table 4.1.: Frequency distribution of *F1-Scores* obtained using the test set. The third column indicates the frequency as a percentage of the total frequencies, indicating the most commonly observed *F1-Score*. The last column indicates the cumulative total.

F1	Freq	% Total	% Cum.
0	2422	25.65	25.65
0.33	13	0.14	25.79
0.4	5	0.05	25.84
0.5	18	0.19	26.03
0.57	1	0.01	26.04
0.67	156	1.65	27.70
1	6827	72.30	100.00

Table 4.2.: Frequency distributions for the number of objects in the frame. The table on the left shows the corresponding frequencies of ground truth (gt_count) detections and system detections (s_count) in the table on the right. Most ground truth and system frames only contain one object per frame as shown by the highlighted rows.

gt_count	Freq	% Total	% Cum.
0	203	2.15	2.15
1	9008	95.40	97.55
2	178	1.89	99.44
3	35	0.37	99.81
4	7	0.07	99.88
5	11	0.12	100.00

s_count	Freq	% Total	% Cum.
0	927	9.82	9.82
1	8309	88.00	97.82
2	193	2.04	99.86
3	13	0.14	100.00

To study the effect of the environmental factors, we aim to model the frame outcome (*F1-Score*) using the environmental factors present in the frame as predictors. We recode the *F1-Scores* using the classification criteria shown in Equation 4.10.

$$RC = \begin{cases} \text{Failure} & \text{F1} == 0 \\ \text{Partial} & 0 < \text{F1} < 1 \\ \text{Perfect} & \text{F1} == 1 \end{cases} \tag{4.10}$$

The GT presenter motion is calculated using the x coordinate of the bounding box centroid between frames A and B in Figure 4.4. The distance is then expressed as a proportion of the frame width to standardise the value across varying resolutions. These continuous values are then split using the median to represent low and high motion categories, with examples shown in Figure 4.7b.

Using the *brms* package [15] in R [6], we fitted a Bayesian Multivariate Multinomial Logistic Regression Model using the Categorical distribution for the following environmental factors:

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

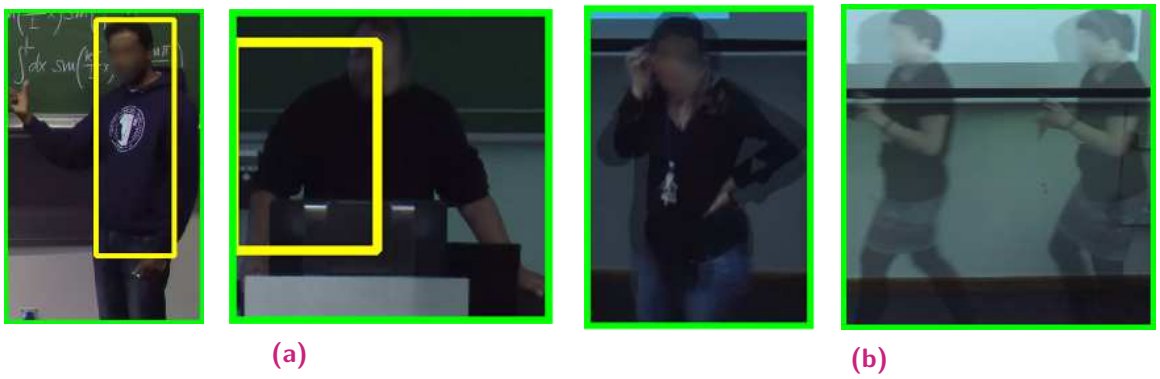


Figure 4.7.: (a) Example of $IOU = 0.3$ between ground truth (green) and system (yellow) detections. (b) Examples of "low" and "high" presenter motion, respectively.

Lighting, *Student Heads*, *Multiple Objects* and *Presenter Motion*. The model fit was satisfactory, and additional diagnostic data can be found in Appendix B. Our data was generated as part of this project, and hence there was no background information about the model parameters prior to observing the data. We, therefore, use non-informative (*brms* default) prior distributions in our model specification and this is not expected to have a negative impact when using a large dataset [50, 86]. The testing dataset was used for the modelling, and to include the presenter motion, we needed to remove all frames where the presenter motion is absent (true negatives). Since we only consider the distance of the primary presenter, we cannot calculate a distance where there is no primary presenter present; hence, these frames needed to be excluded from the modelling only. This difference will be visible when comparing the data in Table 4.1 (before removal) and Table 4.5 (after removal). We consider true negative detections to be perfect and have an *F1-Score* $= 1$.

Figure 4.8 shows the model output and the decision on the null hypothesis (H0) with failure as the reference category. The region of practical equivalence (ROPE) is shown by the blue shaded column and is defined using the conventional range $[-0.1; 0.1]$. The highest density interval (HDI) represents the 95% most credible parameter values, and we reject the null hypothesis (parameter does not influence the outcome) using the HDI+ROPE rule [51]. The rule rejects the null hypothesis when the 95% HDI is not overlapping the ROPE region. We also compute the odds ratios shown in Table 4.3.

In the table, the odds ratio (estimate column) indicates the likelihood of observing an outcome (perfect or partial in comparison to a failure outcome) given the particular condition (lighting, multiple objects, student heads and presenter motion). An odds ratio of one indicates that the condition does not affect the odds of the outcome. A value greater than one indicates greater odds of observing the outcome under the given condition. A value less than one indicates a lower odds of observing the outcome under the given condition.

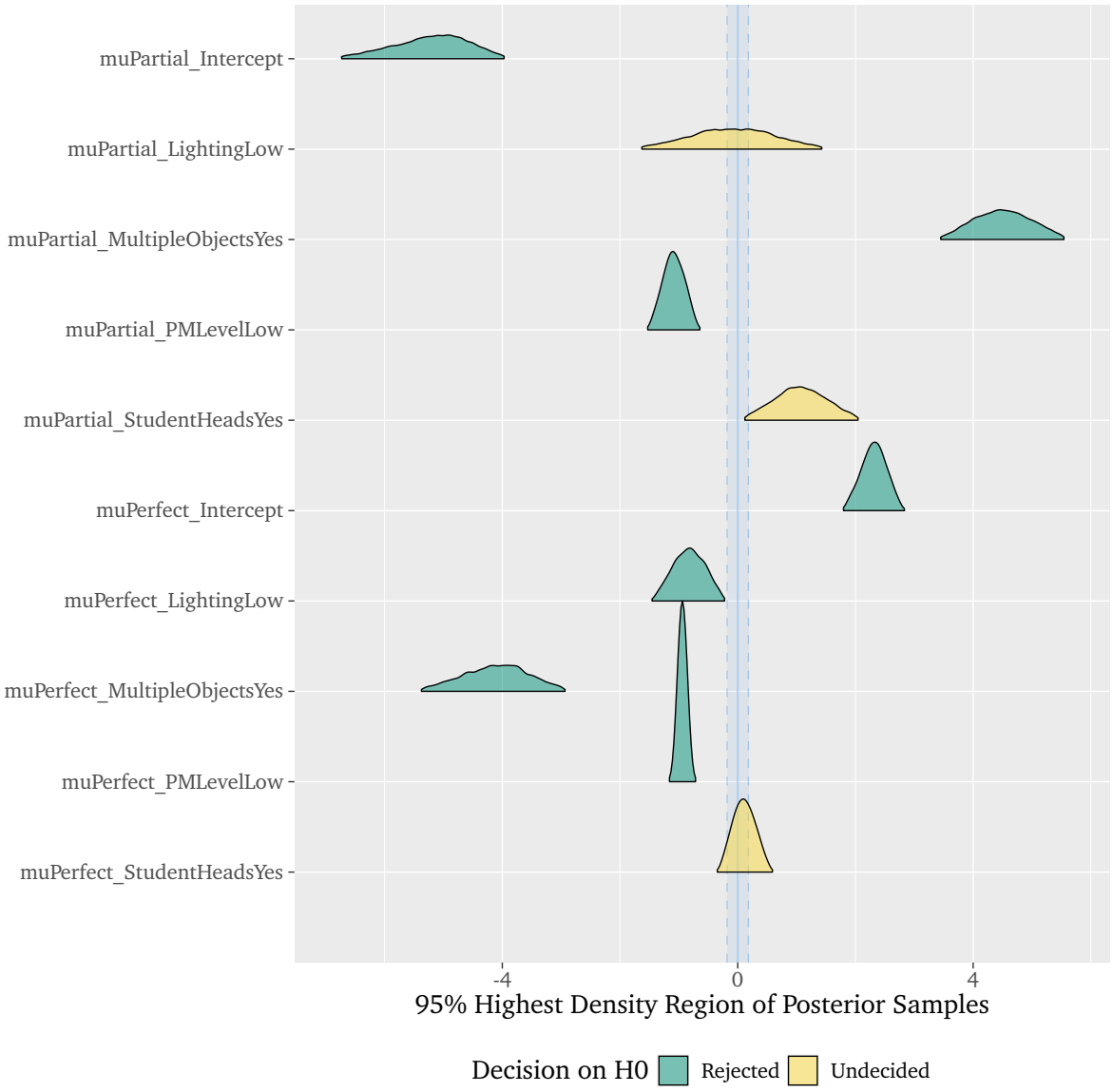


Figure 4.8.: Decision on presenter detection parameters using the HDI+ROPE rule.

Table 4.3.: Odds ratios (shown in the estimate column) for the different presenter detection environmental factors (shown in the first column).

	Estimate	Est.Error	Q2.5	Q97.5
muPartial_Intercept	0.01	2.06	0.00	0.02
muPerfect_Intercept	10.12	1.27	6.27	16.04
muPartial_LightingLow	0.93	2.19	0.20	4.42
muPartial_MultipleObjectsYes	92.23	1.71	33.81	280.24
muPartial_StudentHeadsYes	2.92	1.63	1.15	7.79
muPartial_PMLevelLow	0.34	1.22	0.23	0.50
muPerfect_LightingLow	0.44	1.34	0.25	0.79
muPerfect_MultipleObjectsYes	0.02	1.86	0.00	0.05
muPerfect_StudentHeadsYes	1.11	1.23	0.74	1.68
muPerfect_PMLevelLow	0.39	1.07	0.34	0.45

Each parameter discussed below is compared to the reference category, *failure*, while other parameters remain constant. From the data shown in Figure 4.8 and Table 4.3 we observe the following. The intercepts indicate that in good scenes (good lighting, single object, no student heads and high presenter motion), the odds of observing a partial outcome are almost zero (or 99% less likely), and the odds of observing a perfect outcome is 10.12 times more likely when compared to a failure outcome. The filter module greatly improves the overall *F1-Score* from 0.53 to 0.76 as shown in Table 4.6, suggesting the module effectively mitigates the weaknesses of the differencing-based detection.

We observe that low lighting was not meaningful in predicting a partial outcome over a failure (HDI is contained inside ROPE). However, the odds of observing a perfect outcome under low lighting decreased by 56%. The overall *F1-Score* for each parameter can be seen in Table 4.6 and in low lighting the overall *F1-Score* $= 0.69$ and *F1-Score* $= 0.79$ in good lighting. The contingency table 4.4 shows that the 82.4% of low lighting occurs when the projector is used, which is expected. This finding suggests that in scenes with low lighting conditions, the accuracy of the presenter's location is reduced.

In scenes with multiple objects, the detection does not achieve a perfect detection (98% less likely than failure) and an extremely large odds ratio of 92.23 for observing a partial case. This large value should be interpreted with caution due to the wide credibility interval that is most likely due to separation [58] as a result of the partial category only making up 2.1% of the dataset as shown in Table 4.5. As discussed previously, this is due to the nature of lectures, where there is typically only one presenter in view.

The overall *F1-Score* decreases from 0.77 to 0.42 (see Table 4.6) in the presence of multiple objects, which is the factor with the highest reduction in overall *F1-Score*. Scenes with multiple objects are challenging for our approach, and the filter module only showed a 0.03 improvement.

Table 4.4.: Contingency table comparing projector usage (yes or no) vs. lighting condition (good or low) in frames. Row and column percentages are shown to aid interpretation. In 82.4% of frames, low lighting coincides with projector usage.

	Projector usage		
Lighting	No	Yes	Total
Good	152	4	156
row %	97.4%	2.6%	67.8%
col %	92.1%	6.2%	
Low	13	61	74
row %	17.6%	82.4%	32.2%
col %	7.9%	93.8%	
Total	165	65	230
	71.7%	28.3%	

We compared the continuous run-lengths of frame outcomes (failure, partial and perfect) across all environmental factors, and the distribution of these lengths is shown in Figure 4.9. Failure and partial frames have shorter continuous durations than perfect frames, indicating that failure events are more likely transient events such as students entering or leaving the venue.

Table 4.5.: Contingency table comparing the frame outcomes (failure, partial or perfect detection) in the presence or absence of multiple objects.

	Multiple Objects		
Outcome	No	Yes	Total
Failure	2217	90	2307
row %	96.1%	3.9%	25.1%
col %	24.7%	39.0%	
Partial	75	118	193
row %	38.9%	61.1%	2.1%
col %	0.8%	51.1%	
Perfect	6670	23	6693
row %	99.7%	0.3%	72.8%
col %	74.4%	10.0%	
Total	8962	231	9193
	97.5%	2.5%	

The student heads parameter HDI was contained in the ROPE region and hence was not meaningful in determining the outcome. Looking at Figure 4.8, the HDI is almost central around 0, so it does not have a significantly negative impact on the outcome, and it may lean towards more partial outcomes, however, we cannot confidently say this due to it being included in the ROPE. This result suggests that the filter module is correctly removing these small distractors and is further supported by observing the *F1-Score* improvement shown in Table 4.6 when the filter module is enabled.

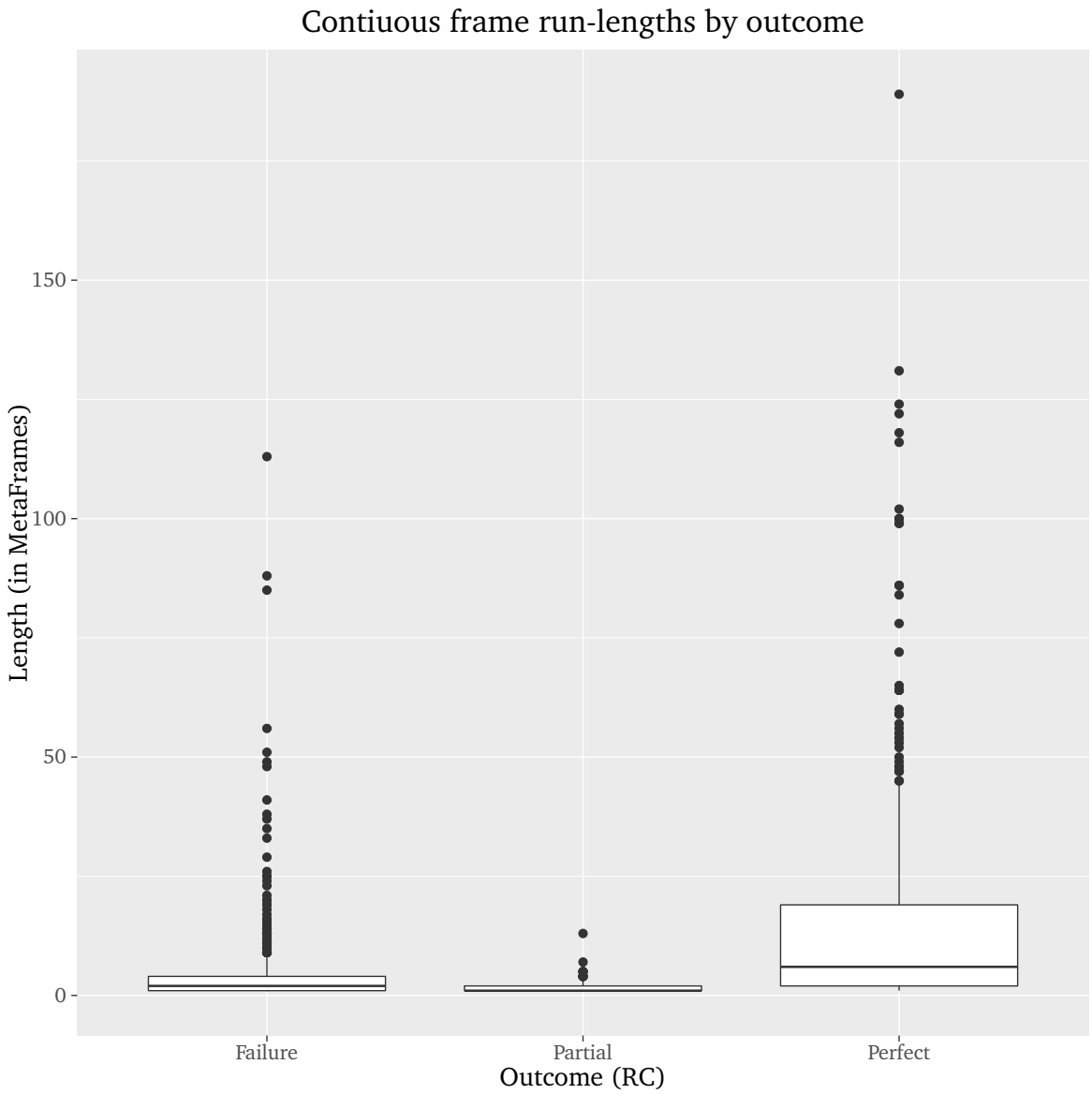


Figure 4.9.: Boxplot showing continuous *MetaFrame* run-lengths by outcome indicating that failure events are typically short in duration.

Table 4.6.: Overall *F1-Score* for the presenter detection under various environmental factors reported individually. For example, looking at frames with low lighting includes all values of other factors. Shading from red to green highlights cases where the filter module can significantly improve results starting below 0.5 without the filter module on.

Factor	Value	F1 (filter off)	F1 (filter on)
Overall	All	0.53	0.76
Lighting	Low	0.38	0.69
	Good	0.62	0.79
Student Heads	No	0.57	0.75
	Yes	0.47	0.77
Multiple Objects	No	0.54	0.77
	Yes	0.39	0.42
Presenter Motion	Low	0.34	0.66
	High	0.74	0.86

Lastly, low presenter motion negatively impacted the outcome, with the likelihood of a perfect outcome decreasing by 61% and 66% for a partial outcome, respectively, compared to failure. In terms of *F1-Score*, there was a reduction from *F1-Score* $= 0.87$ to *F1-Score* $= 0.66$ with low presenter motion.

Overall, the implemented method performs poorly before adding the filtering heuristics. The greatest weakness of the chosen approach was the presence of multiple objects; repairing this is impractical since the differencing does not distinguish separate objects. In general, we do not require high object localisation accuracy, and a visual assessment showed that an overall *F1-Score* $= 0.76$ was still acceptable in terms of localisation correctness (see for example Figure 4.7).

4.6.2 Gesture Detection

Early testing and analysis of our gesture detection module indicated a high false positive rate, which is unreliable for usage in our context. The silhouettes produced by the temporal frame differencing were not consistently well defined and resulted in poor performance by our chosen gesture detection method. As a result of these preliminary findings, we did not proceed with a formal evaluation here.

4.6.3 Writing Detection

Table 4.7 shows the overall single performance values calculated as the average across all quality constraint values, as defined by the authors [93]. Similarly, Table 4.8 and Table 4.9 show the

overall single performance values under various environmental conditions in frames containing only projector content and handwriting, respectively. Frames containing diagrams produce a low *F1-Score* for both projector (*F1-Score* $= 0.41$) and handwritten text (*F1-Score* $= 0.48$). Projector text produces a better detection accuracy (*F1-Score* $= 0.73$) compared to handwriting (*F1-Score* $= 0.60$) in frames where no diagrams are present. This finding can be expected since EAST was not specifically trained to detect handwritten text.

There were no frames in our dataset with good lighting conditions for only projector content (no handwriting present). This observation is similar to that shown in Table 4.4 with a caveat that it did not explicitly exclude projector content frames that also contain handwriting, hence there are four frames in the good lighting and projector usage combination. Low lighting is typical and expected in frames where projector content is present. Low lighting on frames with only handwriting produced an extremely low accuracy, *F1-Score* $= 0.08$.

Reduced visibility is an umbrella factor for three annotated environmental factors — poor visibility, poor contrast and shadows. These annotation labels are grouped together since they are all indicators of the visibility quality of content on the boards/screens. On frames without reduced visibility, projector text has a higher detection accuracy (*F1-Score* $= 0.73$) than handwriting (*F1-Score* $= 0.64$). Under reduced visibility, both performed poorly with projector text being the worst (*F1-Score* $= 0.36$) and handwriting slightly better (*F1-Score* $= 0.42$).

The results obtained seem low in contrast to the results obtained by the presenter detection. A possible reason for this is that, as defined by the authors, these values are an average across all quality constraints. The overall performance graphs in Figure 4.11 indicate that most of the detected boxes are smaller than the GT boxes. The left plot shows the precision, recall and *F1-Score* that depend on the area recall constraint (t_r) while fixing the precision constraint $t_p = 0.4$. Similarly, the right plot varies the precision constraint t_p while fixing the recall constraint $t_r = 0.8$. We used the same thresholds used by the authors [93] as discussed previously.

Table 4.7.: Single overall performance value for handwriting and projector text combined and under all environmental conditions.

Precision	Recall	F1
0.61	0.50	0.55

A recall of $53\% - 71\%$ is achieved ($t_p = 0.4$ and $0 < t_r < 0.7$) and only declines rapidly after about 70% of the precision constraint (t_p) is reached (see Figure 4.11a). As $t_r \to 1$, object recall does not reach 0 and shows a continuous decrease that is almost linear until $t_r = 0.8$ then decreases rapidly thereafter. This indicates that the full GT area is not being detected for each rectangle.

Table 4.8.: Single performance values for writing detection under various environmental factors containing only projector content. There were no frames containing projector content under good lighting in our dataset, indicated by the dashes "—" in the table above.

Factor	Value	Precision	Recall	F1
Overall	NA	0.60	0.57	0.59
Diagrams	No	0.75	0.71	0.73
	Yes	0.43	0.40	0.41
Low Lighting	No	—	—	—
	Yes	0.60	0.57	0.59
Reduced Visibility	No	0.71	0.74	0.73
	Yes	0.40	0.33	0.36

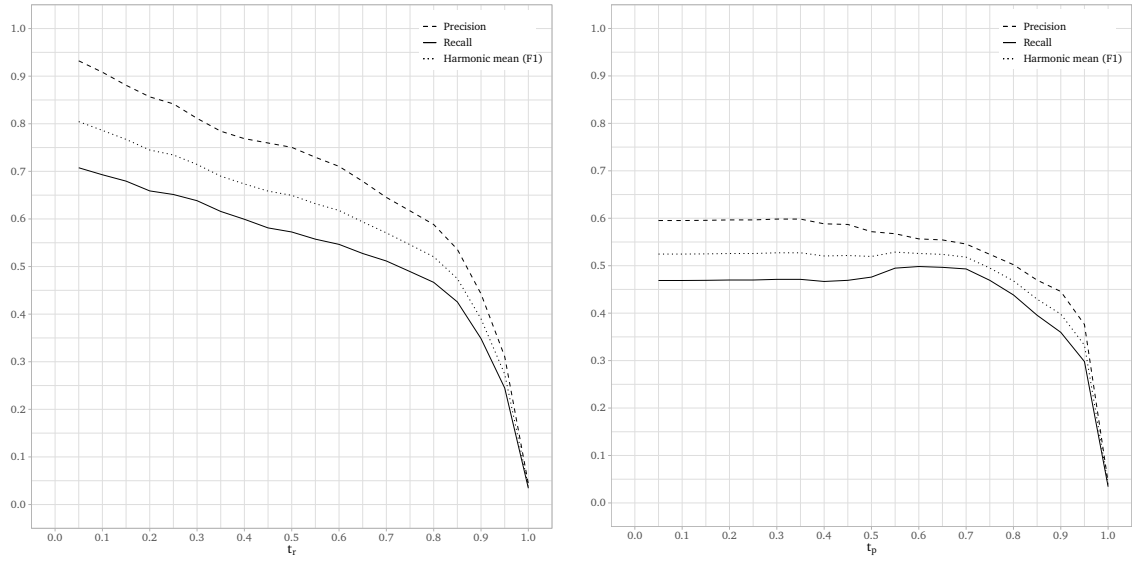
Table 4.9.: Single performance values for writing detection under various environmental factors containing only handwritten content.

Factor	Value	Precision	Recall	F1
Overall	NA	0.62	0.50	0.56
Diagrams	No	0.64	0.56	0.60
	Yes	0.58	0.41	0.48
Low Lighting	No	0.63	0.51	0.57
	Yes	0.09	0.07	0.08
Reduced Visibility	No	0.71	0.58	0.64
	Yes	0.47	0.38	0.42

Figure 4.10.: Ground truth is shown with the green rectangle and system with yellow. As shown in the top three images, poor visibility and diagrams negatively affect the detection quality. Good detections and fragmentation can be seen in the lower three images.

This observation of oversized GT rectangles suggests that the method used for annotating the GT was too loose. As previously discussed, our GT data was labelled with the tightest fitting bounding box around all text on each board (i.e. one rectangle per board, which tightly bounds all the text on that board). This behaviour is more clearly observed when text visibility is reduced, or diagrams are present (see Figure 4.10). Further discussion and additional performance graphs are available in Appendix C and show the performance under varying environmental factors.

The EAST algorithm does not support the detection of drawings, images and other non-textual content or is detected very poorly when small text labels are present. An example of a GT box that will lead to failure can be seen in Figure 4.3b.



(a) (b)

Figure 4.11.: Overall performance graphs of the writing detection module. (a) varying area recall constraint t_r while fixing $t_p = 0.4$ and (b) varying area precision constraint t_p while fixing $t_r = 0.8$.

To further explore the association between the outcome (*F1-Score*) and the environmental factors (*lighting*, *diagrams*, *projector text* and *visibility*) a Bayesian Multivariate Multinomial Logistic Regression Model using the Categorical distribution was used. The model achieved satisfactory convergence, and additional model fit diagnostics can be found in Appendix B. Similar to modelling the presenter environmental effects, we recode the writing detection *F1-Scores* using the same classification criteria shown in Equation 4.10. The modelling was performed using the testing dataset, and we considered true negative detections to be perfect with an *F1-Score* $= 1$.

We use the same HDI+ROPE rule discussed previously to interpret the model output shown in Figure 4.12. The odds ratios can also be seen in Table 4.10. The following discussion on the odds ratios is based on the reference category being "failure". Low lighting resulted in 99% less likelihood of a partial outcome, suggesting that failure was more likely in low lighting conditions. The effect of low lighting on a perfect outcome is undecided since the HDI is included in the ROPE region.

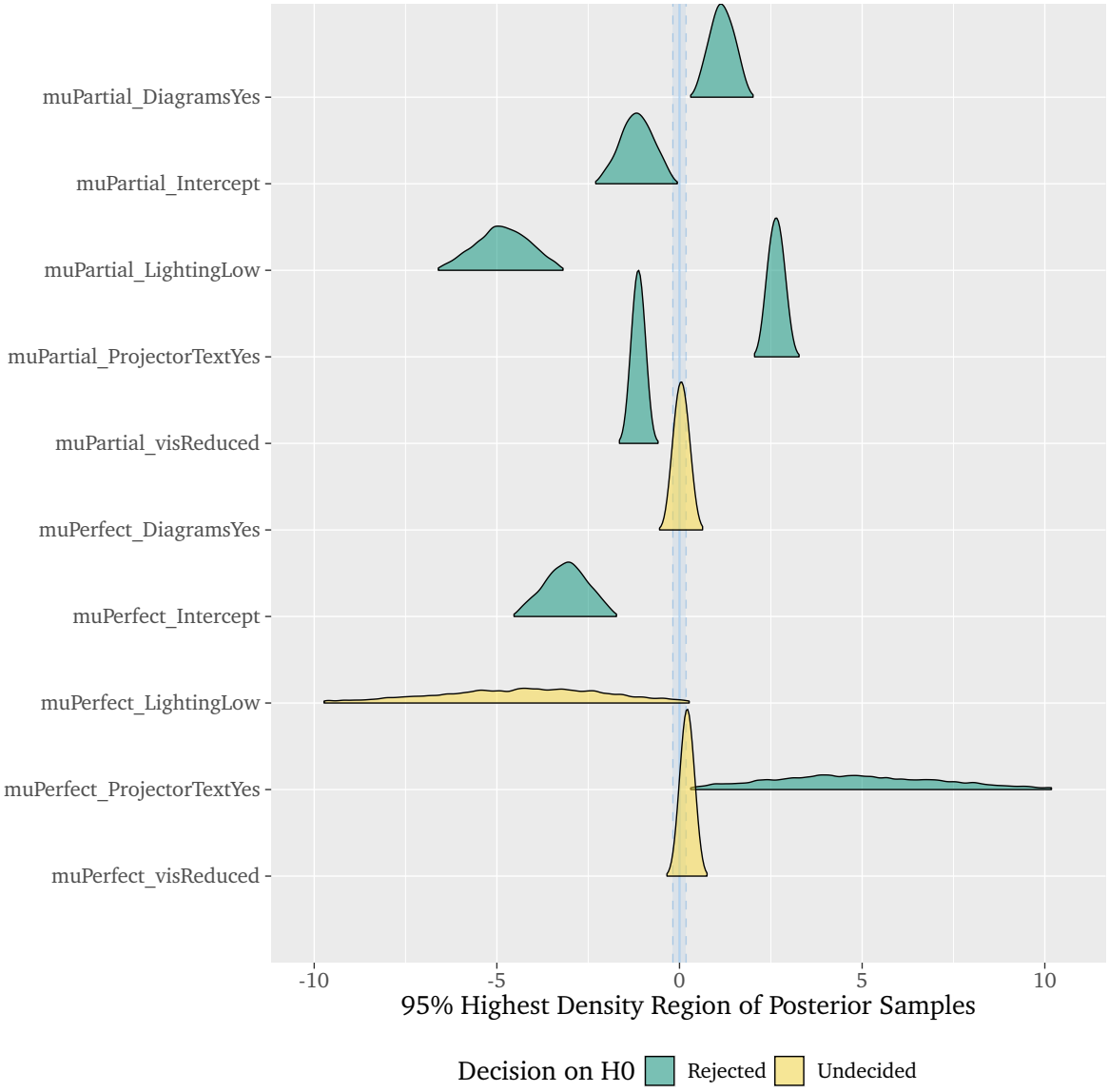


Figure 4.12.: Decision on writing detection parameters using the HDI+ROPE rule.

Projector text produces a very large odds of observing a perfect outcome, however, due to the extremely wide credibility interval, the result should be interpreted with caution. The odds of a partial outcome were 14.04 times higher than a failure, suggesting that projector

Table 4.10.: Odds ratios (estimate column) for various environmental factors (first column) considered in writing detection.

	Estimate	Est.Error	Q2.5	Q97.5
muPartial_Intercept	0.31	1.65	0.11	0.81
muPerfect_Intercept	0.04	1.97	0.01	0.17
muPartial_LightingLow	0.01	2.29	0.00	0.04
muPartial_ProjectorTextYes	14.04	1.24	9.24	21.30
muPartial_DiagramsYes	3.21	1.43	1.61	6.47
muPartial_visReduced	0.32	1.17	0.24	0.44
muPerfect_LightingLow	0.01	13.87	0.00	1.14
muPerfect_ProjectorTextYes	143.38	13.79	1.56	36525.75
muPerfect_DiagramsYes	1.05	1.22	0.71	1.54
muPerfect_visReduced	1.24	1.19	0.89	1.73

text results in a more positive outcome, corroborating our earlier finding and discussion that EAST was not trained specifically for detecting handwritten text. The *brms* tool detected possible multicollinearity between *muPartial_ProjectorTextYes* and *muPartial_LightingLow* ($r = 0.85$), *muPerfect_ProjectorTextYes* and *muPerfect_LightingLow* ($r = 0.85$). This is an interesting observation that also corroborates our previous finding and is due to the presenter typically turning the lights off when using the projector.

A partial outcome was 3.21 times more likely than a failure when diagrams were present, however, the perfect category was inconclusive under this condition. Finally, a reduction in visibility resulted in a higher failure chance than the partial category.

In summary, we observed an overall *F1-Score* $= 0.55$ for the writing detection module. As noted by Wolf and Jolion [93], this *F1-Score* provides satisfactory results. The EAST network is trained to detect text; hence, the improvement with projector text and the degradation with diagrams is expected. Projector text is typically high contrast and consistent, unlike handwriting on dusty boards and is expected to perform better. The detected rectangles were smaller than most of the GT rectangles on frames where diagrams were present or the visibility was reduced. This was exacerbated due to the limitations of the annotated GT data, however, it does indicate that the approach degrades with figures or poor visibility.

The writing information is intended as supplementary input to a VC to enhance framing decisions.

4.6.4 Testing with a VC

We integrated our front-end detection module into a full pipeline with an external VC and report the runtime and file size reduction here. The full pipeline consists of three core components: our front-end module, the VC module and *Cropvid*. The VC module processes our data and produces a list of cropping coordinates. These coordinates are then passed to an adapted version of the original *Cropvid* [7] program that uses the *ffmpeg* [8] filters for cropping and rescaling the video frames.

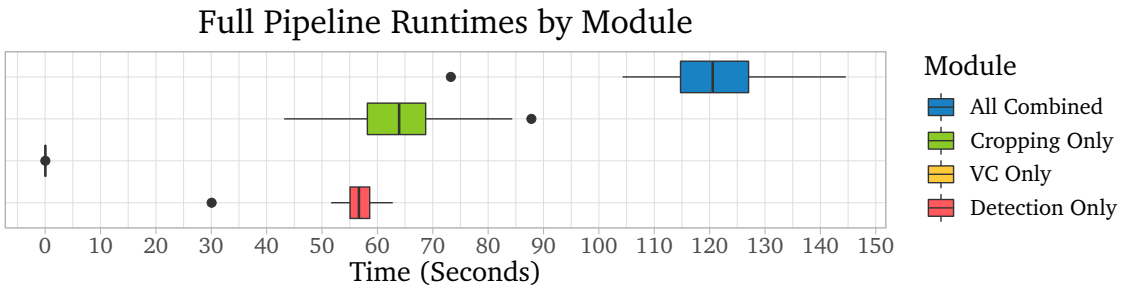


Figure 4.13.: Distribution of runtime per module and the combination of them all on a sample set ($n = 88$) of 2-minute video clips.

Figure 4.13 shows a box plot of runtimes for each of the modules in the full pipeline (produced from a smaller sample of videos, $n = 88$) The VC has an extremely fast runtime since it only reads in a JSON file and generates/exports cropping rectangles, and does not read in video frames, avoiding the expensive disk IO and decoding operations. The detection and cropping modules process video frames and a large portion of time is used for disk IO and decoding operations as previously shown. 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 ≈ 1.5 times the input video length). The median total runtime was just over 2-minutes (i.e. can process a full video in approximately the same time as the input video length).

We also tested the reduction in file size 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 size (on average).

Our tests showed that in the event of detection failure the VC was able to recover quickly and in the presence of multiple objects was able to gracefully zoom out to a wider field of view. In these instances, quality will temporarily be reduced since the 4K frame will be resized to the selected output size of the VC (720p in this case). For example, a fully zoomed-out view is essentially the 4K resolution, however, if the target VC resolution is set to 720p, downscaling

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

will automatically happen in the *Cropvid* process. Lastly, a visual assessment showed that the degradation of writing detection had a minimal impact on the VC output, and in such cases, the VC was capable of resorting to a presenter-centric view.

4.7 Summary

This chapter presented the evaluation methodology and results. The IOU between GT and S rectangles was used to calculate our system's accuracy (*F1-Score*). The presenter detection results indicated an overall *F1-Score* $= 0.53$ and improved to *F1-Score* $= 0.76$ when enabling our filter module. The presence of multiple objects degraded the accuracy, however, these events were shown to be transient and insignificant in our context.

Our writing detection module obtained an overall *F1-Score* $= 0.55$. Handwriting under low lighting conditions had an extremely low accuracy (*F1-Score* $= 0.08$), while frames with reduced visibility affected projector and handwritten content with an accuracy of *F1-Score* $= 0.36$ and *F1-Score* $= 0.42$, respectively.

Our runtime analysis showed that a 720p resolution provided the best overall results and an 89.9% decrease in runtime compared to the 4K frame. A large majority (89%) of the runtime is consumed by the unavoidable operation of reading the video files from the hard drive. For a two-minute video clip, it took an average of 1.4 and 4.58 seconds for presenter and writing detection, respectively. Lastly, we tested our module in a full pipeline with an external VC and observed an average runtime of less than 1.5 times the input video duration.

5 Conclusions

Recording lectures with 4K cameras are an affordable option to capture a high-resolution view of board/screen content and presenter facial expressions. Unfortunately, 4K video files are very large, and the resources required to manage and distribute a large number of such large video files pose financial obstacles to both the institution and the student.

Our project investigates three aspects — presenter, gesture and writing detection — to be used in a larger pipeline with a VC. The aim is to reduce the file size of the 4K video by segmenting a smaller context-centred view while preserving writing clarity. The VC uses our data to form a context and extracts a much smaller fixed-size cropping window containing only relevant information such as the presenter and actively used boards. Our results show that this approach significantly reduces the original 4K video size by approximately 81% while retaining the clarity of writing on the boards/screens. Processing a 2-minute video using the full pipeline took at most 145 seconds (or ≈ 1.5 times the input video duration).

The remainder of this chapter summarises the findings relating to the presenter, gesture and writing detection modules before discussing the limitations of the project and suggesting areas for future work.

5.1 Presenter Detection

Modern object detection techniques utilise machine learning to provide robust detections. Unfortunately, training the network is a computationally intensive and time-consuming process. Similarly, deploying the network is generally computationally expensive, and the best inference times are only achieved using a GPU. Instead, we developed a computationally low-cost object detection module using a classical frame differencing approach and a set of heuristics to mitigate the shortcomings of the differencing approach.

To further optimise the frame differencing, we explored the impact of skipping frames and downscaling the input video frames. We used the training dataset to calibrate these system parameters by analysing the accuracy (*F1-Score*) and runtime. We found that skipping 28 frames yielded the best *F1-Score* and runtime. Furthermore, downscaling the input video from

4K to 720p before analysis resulted in an 89.9% runtime reduction for the motion detection module and produced the best accuracy (*F1-Score* $= 0.8$). Our video reader module consumes the largest portion (89%) of the overall runtime, mostly comprised of disk IO and decoding operations. These disk IO operations are unavoidable, and using a solid-state drive (SSD) could reduce this time.

The test dataset was reserved for testing the system using the calibrated parameters. With the filter module enabled, a similar accuracy was observed in the test dataset (overall *F1-Score* $= 0.76$). This result was a 43.4% improvement on the accuracy from *F1-Score* $= 0.53$ without the filter module enabled. Low presenter motion was a known obstacle for the differencing approach, however, the filter module was shown to almost double (94.1% increase) the accuracy for frames with low presenter motion. The filter was not developed to improve busy scenes with multiple objects due to this being a complex problem of identifying and selecting the correct object. The reduced accuracy on frames with multiple objects was shown to be caused primarily by transient events like students entering/leaving the venue or board motion. Based on our testing, we do not expect these events to have a substantial impact on the VC.

In summary, we revisit and answer our research questions pertaining to presenter detection from Chapter 1.

1. **How accurately can an inexpensive image differencing algorithm detect a presenter in a lecture venue in a post-processing approach?** — *Our results show a high overall accuracy with* F1-Score $= 0.76$ that is fit for our purpose.
2. **How do environmental factors such as lighting and the number of targets in view influence the tracking accuracy?** — *Low lighting and presenter motion negatively impact the approach but were improved sufficiently using the filter heuristics. While multiple objects remain an obstacle to the approach, they are transient and not a concern for our expected use case.*

5.2 Gesture Detection

We explored the detection of large pointing gestures as an extension to the chosen presenter detection approach. We implemented an algorithm that uses the spatial distribution of difference pixels, similar to that suggested by Wulff and Fecke [95] as future work. The difference image's pixel distribution is used assuming that the presenter's body will be skewed to the side of the bounding box during a pointing gesture. This method was found to work in cases where the differencing produced a well-defined silhouette. Low presenter motion resulted in fragmented

silhouettes that complicated the search for gestures. Furthermore, while using a larger skip frame value produced better presenter detection results and faster runtime, it also distorted silhouettes. This approach should work in environments with uniform background colours like a green screen. However, given the range of unconstrained backgrounds, even after applying our heuristics, our frame differencing approach was deemed unsuitable for producing the quality of silhouettes required for this gesture detection method.

In relation to our research question for this module: **How accurately can we detect the presence and direction of gestures as an extension to the frame differencing?** — Frame differencing techniques are not suitable for gesture detection.

5.3 Writing Detection

Unlike presenter detection, writing is added slowly over time, and the detection algorithm can run less frequently. This enabled the use of EAST, a machine-learning-based text detector that offers robust text detection compared to its classical counterparts.

Similar to presenter detection, we calibrated the system parameters for writing detection using the training dataset and analysed the accuracy (*F1-Score*) and runtime. The module used the same 720p resolution used by the presenter detection and provided the best accuracy. At this resolution, and performing a detection at 30 second intervals, the writing detection module had a mean runtime of 4.58 seconds for a 2-minute length video (or 3.8% of the input video duration).

Below we revisit our research questions for this module and answer them.

1. **How accurately can the EAST detector using the pre-trained neural network detect writing on boards in a lecture venue?** *An overall* F1-Score $= 0.55$ *was achieved using the test set — these results were comparable to the results presented in the work of Wolf and Jolion [93].*

2. **How do environmental factors such as lighting, writing visibility, diagrams/images and protector text or handwriting influence the detection accuracy?** *Our data shows that low lighting severely affects the detection accuracy for handwriting (*F1-Score $= 0.08$*) while projector text performs better (*F1-Score $= 0.59$*). Diagrams decrease the accuracy from* F1-Score $= 0.73$ *to* F1-Score $= 0.41$ *and* F1-Score $= 0.60$ *to* F1-Score $= 0.48$ *for projector content and handwriting respectively. This observation is expected as detecting diagrams is a known limitation since EAST is a text detector.*

The writing information is intended as supplementary input to a VC to enhance framing decisions, and the accuracy achieved is expected to be sufficient for our purpose. ***Our testing showed that the degradation of writing detection had a minimal impact on the VC output, and in such cases, the VC was capable of gracefully resorting to a presenter-centric view.

5.4 Limitations and Future Work

The differencing approach produced satisfactory results for the purpose of presenter detection but was insufficient for the chosen gesture detection approach. Selecting a larger skip frame is preferable for presenter detection as it improves detection when presenter motion is low or stationary. While this generates an oversized bounding box when the presenter walks, this is better than having no bounding box due to a small skip frame value. Unfortunately, both extremes (small or large skip frames) produced fragmented bounding boxes due to holes in the silhouettes. We have two suggestions for future work in this regard. One suggestion would be to explore the computational cost and efficacy of additional post-processing heuristics such as an adaptive morphology operation and flood fill to improve the silhouette.

An alternative approach would be to reconsider using a deep-learning-based pose detector (see the work of Cao et al. [16]) that could serve both purposes, presenter and gesture detection. Optimisation of this approach needs careful consideration. While using a lower detection frequency (similar to our approach for writing detection) may be sufficient for presenter detection, if it is too infrequent, it could miss the gesture entirely. On the other hand, running a machine-learning-based detection at a frequency suitable for gesture detection may not yield the same runtime efficiency as our current frame differencing implementation.

The writing detection module currently uses the same frame resolution as the presenter detection. Future work could explore a self-calibrating approach by periodically running EAST with a set of standard resolutions (up to and including the *workingDimension*) and evaluating and selecting the resolution providing the best word count for the computational time trade-off. Additionally, the EAST network can be trained on a dataset that includes handwriting to see if this would improve detection accuracy. Lastly, the chosen annotation approach for the writing only provided a single tightest bounding box per board since annotating over 350000 frames in such detail would have been unrealistic. The current approach's limitation is that the system rarely detects the whole GT region, resulting in seemingly lower accuracy. Future work could annotate a smaller sample of frames with greater detail. Lastly, other approaches can incorporate non-textual detection support, including images and diagrams.

The software developed as part of this project was an extension to an existing open source project, *TRACK4K*, and will also be made open source to enable future development and improvement. The source code will be available under the same GitHub profile: https://github.com/LectureTracking.

Bibliography

[1] G. D. Abowd et al. “Teaching and learning as multimedia authoring: the classroom 2000 project”. In: *Proceedings of the Fourth ACM International Conference on Multimedia*. MULTIMEDIA ’96. Boston, Massachusetts, USA: ACM, 1997, pp. 187–198.

[2] A. S. Abutaleb. “Automatic thresholding of gray-level pictures using two-dimensional entropy”. In: *Computer Vision, Graphics, and Image Processing* 47.1 (July 1989), pp. 22–32.

[3] S. Arseneau and J. R. Cooperstock. “Presenter tracking in a classroom environment”. In: *25th Annual Conference of the IEEE Industrial Electronics Society*. Vol. 1. IECON’99. San Jose, CA, USA: IEEE, 1999, pp. 145–148.

[4] Y. Baek et al. “Character Region Awareness for Text Detection”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Long Beach, CA, USA: IEEE, June 2019, pp. 9357–9366.

[5] S. R. Balaji and S. Karthikeyan. “A survey on moving object tracking using image processing”. In: *2017 11th International Conference on Intelligent Systems and Control (ISCO)*. Coimbatore, India: IEEE, Feb. 2017, pp. 469–474.

[6] P. Banerjee, U. Bhattacharya, and B. B. Chaudhuri. “Automatic Detection of Handwritten Texts from Video Frames of Lectures”. In: *14th International Conference on Frontiers in Handwriting Recognition*. Hersonissos, Greece: IEEE, Dec. 2014, pp. 627–632.

[7] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. “Performance of optical flow techniques”. In: *International Journal of Computer Vision* 12.1 (Feb. 1994), pp. 43–77.

[8] C. P. Benner and C. A. Rogers. “A new plan for instructing large classes in mathematics by television and films”. In: *The Mathematics Teacher* 53.5 (1960), pp. 371–375.

[9] M. Bianchi. “Automatic Video Production of Lectures Using an Intelligent and Aware Environment”. In: *Proceedings of the 3rd International Conference on Mobile and Ubiquitous Multimedia*. MUM ’04. College Park, Maryland, USA: ACM, 2004, pp. 117–123.

[10] N. B. Bo et al. “Detection of a hand-raising gesture by locating the arm”. In: *2011 IEEE International Conference on Robotics and Biomimetics*. Karon Beach, Thailand: IEEE, 2011, pp. 976–980.

[11] N. Bozorgzadeh and R. J. Bathurst. “Bayesian model checking, comparison and selection with emphasis on outlier detection for geotechnical reliability-based design”. In: *Computers and Geotechnics* 116 (Dec. 2019), p. 103181.

[12] G. Bradski and A. Kaehler. “Overview”. In: *Learning OpenCV 3*. Ed. by D. Schanafelt, K. Brown, and R. Monaghan. O’Reilly Media, Inc., 2016. Chap. 1, pp. 1–19.

[13] C. A. Brooks et al. “OpenCast Matterhorn 1.1: Reaching New Heights”. In: *Proceedings of the 19th ACM International Conference on Multimedia*. MM ’11. Scottsdale, Arizona, USA: ACM, 2011, pp. 703–706.

[14] S. Brutzer, B. Höferlin, and G. Heidemann. “Evaluation of background subtraction techniques for video surveillance”. In: *Computer Vision and Pattern Recognition*. Colorado Springs, CO, USA: IEEE, 2011, pp. 1937–1944.

[15] 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.

[16] Z. Cao et al. “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.

[17] P. Carr, M. Mistry, and I. Matthews. “Hybrid Robotic/Virtual Pan-Tilt-Zom Cameras for Autonomous Event Recording”. In: *Proceedings of the 21st ACM international conference on Multimedia*. MM ’13. Barcelona, Spain: ACM, 2013, pp. 193–202.

[18] H. D. Cheng et al. “Color image segmentation: Advances and prospects”. In: *Pattern Recognition* 34.12 (2001), pp. 2259–2281.

[19] H.-P. Chou et al. “Automated lecture recording system”. In: *2010 International Conference on System Science and Engineering*. Taipei: IEEE, 2010, pp. 167–172.

[20] C. Choudary and T. Liu. “Extracting content from instructional videos by statistical modelling and classification”. In: *Pattern Analysis and Applications* 10.2 (2007), pp. 69–81.

[21] 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.

[22] 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.

[23] J. W. Davis and A. F. Bobick. “The representation and recognition of human movement using temporal templates”. In: *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. San Juan, PR, USA: IEEE, 1997, pp. 928–934.

[24] A. Dhakal, S. G. Kulkarni, and K. K. Ramakrishnan. “Machine Learning at the Edge: Efficient Utilization of Limited CPU/GPU Resources by Multiplexing”. In: *2020 IEEE 28th International Conference on Network Protocols (ICNP)*. Madrid, Spain: IEEE, Oct. 2020, pp. 1–6.

[25] P. E. Dickson, W. R. Adrion, and A. R. Hanson. “Whiteboard content extraction and analysis for the classroom environment”. In: *10th IEEE International Symposium on Multimedia*. Berkeley, CA, USA: IEEE, 2008, pp. 702–707.

[26] Edgar. “The Educational Use of Audible Motion Pictures”. In: *Educational Research Bulletin* 10.11 (1931), pp. 290–294.

[27] M. Ester et al. “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.

[28] L. Fan et al. “A survey on multiple object tracking algorithm”. In: *2016 IEEE International Conference on Information and Automation (ICIA)*. Ningbo, China: IEEE, Aug. 2016, pp. 1855–1862.

[29] M. Fei, J. Li, and H. Liu. “Visual tracking based on improved foreground detection and perceptual hashing”. In: *Neurocomputing* 152 (Mar. 2015), pp. 413–428.

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

[31] D. Gatica-Perez et al. “Audiovisual Probabilistic Tracking of Multiple Speakers in Meetings”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 15.2 (Feb. 2007), pp. 601–616.

[32] 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.

[33] T. Y. Goh et al. “Performance analysis of image thresholding: Otsu technique”. In: *Measurement* 114 (2018), pp. 298–307.

[34] N. L. Greene. “Motion Pictures in the Classroom”. In: *The Annals of the American Academy of Political and Social Science* 128 (1926), pp. 122–130.

[35] A. Handa et al. “Real-Time Camera Tracking: When is High Frame-Rate Best?” In: *European Conference on Computer Vision*. Ed. by A. Fitzgibbon et al. Vol. 7578. ECCV 2012. Berlin, Heidelberg: Springer, 2012, pp. 222–235.

[36] A. Haubold and J. R. Kender. “Analysis and interface for instructional video”. In: *2003 International Conference on Multimedia and Expo*. Baltimore, MD, USA: IEEE, 2003, pp. 705–708.

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

[38] 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.

[39] M. Hossain and M. Jenkin. “Recognizing hand-raising gestures using HMM”. In: *The 2nd Canadian Conference on Computer and Robot Vision (CRV’05)*. CRV’05. Victoria, BC, Canada: IEEE, 2005, pp. 405–412.

[40] W. Hu et al. “A survey on visual surveillance of object motion and behaviors”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 34.3 (Aug. 2004), pp. 334–352.

[41] S. Ilarri et al. “A Friendly Location-aware System to Facilitate the Work of Technical Directors When Broadcasting Sport Events”. In: *Mobile Information Systems* 8.1 (2012), pp. 17–43.

[42] A. S. Imran and F. A. Cheikh. “Blackboard content classification for lecture videos”. In: *18th IEEE International Conference on Image Processing*. Brussels, Belgium: IEEE, Sept. 2011, pp. 2989–2992.

[43] E. Jain et al. “Gaze-Driven Video Re-Editing”. In: *ACM Transactions on Graphics* 34.2 (2015), pp. 1–12.

[44] D. Karatzas et al. “ICDAR 2013 Robust Reading Competition”. In: *12th International Conference on Document Analysis and Recognition*. Washington, DC, USA: IEEE, 2013, pp. 1484–1493.

[45] D. Karatzas et al. “ICDAR 2015 competition on Robust Reading”. In: *13th International Conference on Document Analysis and Recognition*. Tunis, Tunisia: IEEE, Nov. 2015, pp. 1156–1160.

[46] A. Kayom, M. Khairuzzaman, and S. Chaudhury. “Multilevel thresholding using grey wolf optimizer for image segmentation”. In: *Expert Systems with Applications* 86 (2017), pp. 64–76.

[47] D. Keegan. “Teaching and learning by satellite in a European virtual classroom”. In: *Open and Distance Learning Today*. Ed. by D. K. Fred Lockwood. 1st ed. London: Routledge, 1995. Chap. 11, pp. 108–118.

[48] D. Keegan. “The Study of Distance Education”. In: *Foundations of Distance Education*. 3rd ed. Routledge, 2013. Chap. 1, pp. 1–20.

[49] D. Kim et al. “Vision-based arm gesture recognition for a long-range human–robot interaction”. In: *The Journal of Supercomputing* 65.1 (July 2013), pp. 336–352.

[50] C. König and R. Van De Schoot. “Bayesian statistics in educational research: a look at the current state of affairs”. In: *Educational Review* 70.4 (2018), pp. 486–509.

[51] J. K. Kruschke. “Rejecting or Accepting Parameter Values in Bayesian Estimation”. In: *Advances in Methods and Practices in Psychological Science* 1.2 (May 2018), pp. 270–280.

[52] J. S. Kulchandani and K. J. Dangarwala. “Moving object detection: Review of recent research trends”. In: *2015 International Conference on Pervasive Computing (ICPC)*. Pune, India: IEEE, Jan. 2015, pp. 1–5.

[53] M. Kumar et al. “Zooming On All Actors: Automatic Focus+Context Split Screen Video Generation”. In: *Computer Graphics Forum* 36.2 (2017), pp. 455–465.

[54] F. Lampi, S. Kopf, and W. Effelsberg. “Automatic Lecture Recording”. In: *Proceedings of the 16th ACM International Conference on Multimedia*. MM ’08. Vancouver, British Columbia, Canada: ACM, 2008, pp. 1103–1105.

[55] F. Liu and M. Gleicher. “Video Retargeting: Automating Pan and Scan”. In: *Proceedings of the 14th ACM International Conference on Multimedia*. MM ’06. Santa Barbara, CA, USA: ACM, 2006, pp. 241–250.

[56] Q. Liu et al. “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.

[57] Q. Liu et al. “FlySPEC: A Multi-User Video Camera System with Hybrid Human and Automatic Control”. In: *Proceedings of the ACM International Multimedia Conference and Exhibition*. MULTIMEDIA ’02. Juan-les-Pins, France: ACM, 2002, pp. 484–492.

[58] M. A. Mansournia et al. “Separation in Logistic Regression: Causes, Consequences, and Control”. In: *American Journal of Epidemiology* 187.4 (Apr. 2018), pp. 864–870.

[59] J. Martin and J. B. Durand. “Automatic handwriting gestures recognition using hidden Markov models”. In: *Proceedings - 4th IEEE International Conference on Automatic Face and Gesture Recognition*. Grenoble, France: IEEE, 2000, pp. 403–409.

[60] A. Mavlankar et al. “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.

[61] S. Mukhopadhyay and B. Smith. “Passive Capture and Structuring of Lectures”. In: *Proceedings of the Seventh ACM International Conference on Multimedia (Part 1)*. MULTIMEDIA ’99. Orlando, Florida, USA: ACM, 1999, pp. 477–487.

[62] P. Natarajan and R. Nevatia. “Online, Real-time Tracking and Recognition of Human Actions”. In: *2008 IEEE Workshop on Motion and video Computing*. Copper Mountain, CO, USA: IEEE, Jan. 2008, pp. 1–8.

[63] W. Niblack. *An introduction to digital image processing*. 1st ed. Strandberg Publishing Company, 1985, pp. 1–215.

[64] K. Nickel and R. Stiefelhagen. “Visual recognition of pointing gestures for human-robot interaction”. In: *Image and Vision Computing* 25.12 (2007), pp. 1875–1884.

[65] F. V. O’Callaghan et al. “The use of lecture recordings in higher education: A review of institutional, student, and lecturer issues”. In: *Education and Information Technologies* 22.1 (Jan. 2017), pp. 399–415.

[66] S. Ojha and S. Sakhare. “Image processing techniques for object tracking in video surveillance - A survey”. In: *International Conference on Pervasive Computing*. Pune, India: IEEE, Apr. 2015, pp. 1–6.

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

[68] R. Padilla et al. “A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit”. In: *Electronics* 10.3 (Jan. 2021).

[69] Y.-T. Pai, Y.-F. Chang, and S.-J. Ruan. “Adaptive thresholding algorithm: Efficient computation technique based on intelligent block detection for degraded document images”. In: *Pattern Recognition* 43.9 (Sept. 2010), pp. 3177–3187.

[70] A. Ramamoorthy et al. “Recognition of dynamic hand gestures”. In: *Pattern Recognition* 36.9 (2003), pp. 2069–2081.

[71] S. S. Rautaray and A. Agrawal. “Vision based hand gesture recognition for human computer interaction: a survey”. In: *Artificial Intelligence Review* 43.1 (Nov. 2015), pp. 1–54.

[72] J. Rios-Amaya, J. Secker, and C. Morrison. *Lecture Recording in Higher Education: Risky Business or Evolving Open Practice*. Tech. rep. November. University of Kent, 2016, pp. 1–44.

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

[74] C. Saravanan. “Color image to grayscale image conversion”. In: *Second International Conference on Computer Engineering and Applications*. Vol. 2. Bali, Indonesia: IEEE, 2010, pp. 196–199.

[75] J. Sauvola and M. Pietikäinen. “Adaptive document image binarization”. In: *Pattern Recognition* 33.2 (2000), pp. 225–236.

[76] M. F. Savaş, H. Demirel, and B. Erkal. “Moving object detection using an adaptive background subtraction method based on block-based structure in dynamic scene”. In: *Optik* 168 (Sept. 2018), pp. 605–618.

[77] J. Schindelin et al. “Fiji: an open-source platform for biological-image analysis”. In: *Nature Methods* 9.7 (June 2012), pp. 676–682.

[78] S. S. Sengar and S. Mukhopadhyay. “A novel method for moving object detection based on block based frame differencing”. In: *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*. Dhanbad, India: IEEE, July 2016, pp. 467–472.

[79] S. S. Sengar and S. Mukhopadhyay. “Moving object detection using statistical background subtraction in wavelet compressed domain”. In: *Multimedia Tools and Applications* 79.9 (Mar. 2020), pp. 5919–5940.

[80] A. Senthil Murugan et al. “A study on various methods used for video summarization and moving object detection for video surveillance applications”. In: *Multimedia Tools and Applications* 77.18 (Sept. 2018), pp. 23273–23290.

[81] H. V. Shin et al. “Visual Transcripts: Lecture Notes from Blackboard-Style Lecture Videos”. In: *ACM Transactions on Graphics* 34.6 (Oct. 2015), pp. 1–10.

[82] L. Tang and J. R. Kender. “A unified text extraction method for instructional videos”. In: *IEEE International Conference on Image Processing*. Vol. 3. Genova, Italy: IEEE, 2005, pp. 1216–1219.

[83] I. N. Toppin. “Video lecture capture (VLC) system: A comparison of student versus faculty perceptions”. In: *Education and Information Technologies* 16.4 (Dec. 2011), pp. 383–393.

[84] K. Toyama et al. “Wallflower: principles and practice of background maintenance”. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 1. Kerkyra, Greece: IEEE, 1999, pp. 255–261.

[85] B. Urala Kota et al. “Summarizing Lecture Videos by Key Handwritten Content Regions”. In: *2019 International Conference on Document Analysis and Recognition Workshops*. Sydney, NSW, Australia: IEEE, Nov. 2019, pp. 13–18.

[86] R. Van De Schoot et al. “Analyzing small data sets using Bayesian estimation: the case of posttraumatic stress symptoms following mechanical ventilation in burn survivors”. In: *European Journal of Psychotraumatology* 6.1 (2015), pp. 1–13.

[87] M. Wallick et al. “Marker and chalkboard regions”. In: *Proceedings of Mirage 2005 (Computer Vision/Computer Graphics Collaboration Techniques and Applications)*. INRIA, Rocquencourt, France: ACM, 2005, pp. 223–228.

[88] F. Wang, C.-W. Ngo, and T.-C. Pong. “Lecture Video Enhancement and Editing by Integrating Posture, Gesture, and Text”. In: *IEEE Transactions on Multimedia* 9.2 (2007), pp. 397–409.

[89] F. Wang, C.-W. Ngo, and T.-C. Pong. “Simulating a Smartboard by Real-Time Gesture Detection in Lecture Videos”. In: *IEEE Transactions on Multimedia* 10.5 (2008), pp. 926–935.

[90] L. Wen et al. “UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking”. In: *Computer Vision and Image Understanding* 193 (Apr. 2020), pp. 1–20.

[91] M. B. Winkler et al. “Automatic Camera Control for Tracking a Presenter during a Talk”. In: *2012 IEEE International Symposium on Multimedia*. Irvine, CA, USA: IEEE, 2012, pp. 471–476.

[92] T. A. B. Wirayuda et al. “Development methods for hybrid motion detection (frame difference-automatic threshold)”. In: *2013 International Conference of Information and Communication Technology (ICoICT)*. Bandung, Indonesia: IEEE, 2013, pp. 218–222.

[93] 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.

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

[95] 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.

[96] B. Wulff et al. “LectureSight: an open source system for automatic camera control for lecture recordings”. In: *Interactive Technology and Smart Education* 11.3 (2014), pp. 184–200.

[97] Z. Xu, D. Zhang, and L. Du. “Moving Object Detection Based on Improved Three Frame Difference and Background Subtraction”. In: *2017 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII)*. CIICII 2017. Wuhan, China: IEEE, Mar. 2017, pp. 79–82.

[98] K. Yadav et al. “ViZig: Anchor Points Based Non-Linear Navigation and Summarization in Educational Videos”. In: *Proceedings of the 21st International Conference on Intelligent User Interfaces*. IUI ’16. Sonoma, California, USA: ACM, Mar. 2016, pp. 407–418.

[99] J. Yao and J. R. Cooperstock. “Arm gesture detection in a classroom environment”. In: *Sixth IEEE Workshop on Applications of Computer Vision*. Orlando, FL, USA: IEEE, 2002, pp. 153–157.

[100] 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.

[101] A. Yilmaz, O. Javed, and M. Shah. “Object tracking: A survey”. In: *ACM Computing Surveys* 38.4 (Dec. 2006), 13–es.

[102] 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.

[103] C. Zhang and Y. Rui. “Automated lecture services”. In: *Studies in Computational Intelligence*. Ed. by G. A. Tsihrintzis and L. C. Jain. Vol. 120. Springer, 2008, pp. 351–375.

[104] Q. Zhang et al. “Edge Video Analytics for Public Safety: A Review”. In: *Proceedings of the IEEE* 107.8 (July 2019), pp. 1675–1696.

[105] X. Zhao, W. P. Zhou, and H. Z. Shu. “A Novel Method for Human Action Analysis Based on Temporal Template”. In: *2009 First International Conference on Information Science and Engineering*. Nanjing, China: IEEE, 2009, pp. 1083–1086.

[106] X. Zhou et al. “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.

System Pseudocode

A

A.1 Pseudocode for core system modules

The main class acts as the main system loop and is responsible for parsing command line arguments and invoking sub-modules. Pseudocode for this class can be seen in Algorithm 1, followed by pseudocode for the three core modules presenter (Algorithm 2), gesture (Algorithm 3) and writing detection (Algorithm 4).

Algorithm 1: Main loop

```
Function main(command line arguments)
    input  : Input video filename and configuration filename
    output : JSON file containing detected object data

    videoFileName ← ParseCommandLine(args);
    configFileName ← ParseCommandLine(args);

    dataBundle; // Stores parameters and all data generated by sub-modules
    dataBundle ← InitialiseSystemParameters(videoFileName, configFileName);

    while video has frames remaining do
        readVideoFrames(dataBundle); // Read segment of frames into dataBundle

        // Call sub-modules and pass dataBundle object as an argument
        presenterDetection(dataBundle); // See Algorithm 2.
        gestureDetection(dataBundle); // See Algorithm 3.
        writingDetection(dataBundle); // See Algorithm 4.
        releaseMemory(dataBundle); // Discard segment of all stored frames
    DataFilter(dataBundle);
    JSONWriter(dataBundle);
```

Algorithm 2: Presenter Detection

```
Function main(dataBundle)
    input  : dataBundle
    output : dataBundle

    inputFramesArray ← dataBundle.inputFrames;

    for i ← 0 to range (inputFramesArray) do
        // Iterate over segment of frames that were read into memory
        frameA ← inputFramesArray[i];
        frameB ← inputFramesArray[i + 1];

        // Convert both images to greyscale
        frameA_Grey ← greyscale(frameA);
        frameA_Grey ← greyscale(frameB);

        // Perform pixel-wise difference
        frameAB_Binary ← pixelwiseDifference(frameA_Grey, frameB_Grey);

        // Apply thresholding operations
        frameAB_Binary ← thresholdOtsu(frameAB_Binary);

        // Apply morphological operations
        frameAB_Binary ← morphology(frameAB_Binary);

        // Perform connected components analysis (CCA)
        connectedComponentLabels ← connectedComponents(frameAB_Binary);

        // Generate bounding boxes
        boundingBoxArray ← boundObjects(connectedComponentLabels);

        // Merge overlapping bounding boxes
        boundingBoxArray ← mergeBoundingBoxes(boundingBoxArray);

        // Append new MetaFrame (representing data of input frame i and i + 1)
           to dataBundle
        dataBundle ← metaFrame(i, i + 1, boundingBoxArray, frameA, frameB,
         frameAB_Binary);
```

Algorithm 3: Gesture Detection

```
Function main(dataBundle)
    input  : dataBundle
    output : dataBundle

    metaFramesArray ← dataBundle.metaFrames;

    foreach metaFrame in metaFramesArray do
        foreach boundingBox in metaFrame do
            cropImage ← Crop frameAB_Binary using boundingBox;

            // Partition cropImage into four quadrants and count white pixels
               in each quadrant
            topLeftCount ← Number of white pixels in Top Left quadrant;
            bottomLeftCount ← Number of white pixels in Bottom Left quadrant;
            topRightCount ← Number of white pixels in Top Right quadrant;
            bottomRightCount ← Number of white pixels in Bottom Right quadrant;

            leftCount ← (topLeftCount + bottomLeftCount);
            rightCount ← (topRightCount + bottomRightCount);

            totalWhitePixelCount ← ∑(All white pixels);

            topLeftFraction ← (topLeftCount/totalWhitePixelCount);
            bottomLeftFraction ← (bottomLeftCount/totalWhitePixelCount);
            topRightFraction ← (topRightCount/totalWhitePixelCount);
            bottomRightFraction ← (bottomRightCount/totalWhitePixelCount);

            // Check for imbalance between left and right of image
            if ||leftCount − rightCount|| > gestureThreshold then
                if leftCount > rightCount then
                    // Right gesture
                    if topRightFraction < bottomRightFraction then
                        return 0; // Not a gesture

                    // Store right gesture
                    boundingBox ← RightGesture;
                else
                    // Left gesture
                    if topLeftFraction < bottomLeftFraction then
                        return 0; // Not a gesture

                    // Store left gesture
                    boundingBox ← LeftGesture;
```

Algorithm 4: Writing Detection

Function `main`(*dataBundle*)

input : dataBundle
output : dataBundle

metaFramesArray ← dataBundle.metaFrames;
lastTimeStamp ← 0;

foreach $metaFrame$ *in* metaFramesArray **do**
  currentTimeStamp ← metaFrame.frameA timestamp;
  if (currentTimeStamp − lastTimeStamp) ≥ writingDetectionSkipTime **then**
    squareFrame ← Resize image to square ($width == height$) using *workingDimension*

    `// Call EAST - Returns rotated rectangles enclosing each word`
    EASTOutputArray ← `EAST`(squareFrame)

    `// Apply non-maximum suppression to remove overlapping boxes`
    EASTOutputArray ← `NMS`(EASTOutputArray)

    writingGroup ← Recursively cluster rotated rectangles into groups

    `// Append writing groups to metaFrame`
    metaFrame ← writingGroup
  lastTimeStamp ← currentTimeStamp;

Bayesian Model Fit Diagnostics

B

In this appendix, we detail additional information on the model fitting process used in the presenter detection results section 4.6.1 and writing detection results section 4.6.3.

B.1 Presenter Detection

B.1.1 Model Parameters

The leave-one-out cross-validation (LOO) function identified a better model fit using the clip filename vs. parent (full-length video) filename as a random effect. A smaller *leave-one-out information criterion (LOOIC)* value typically indicates a higher predictive accuracy [11]. Listing B.1 showing a smaller *LOOIC* value for the "fit" model which is using the clip file name.

Satisfactory model convergence was achieved ($\widehat{R} = 1$ for all parameters) using 4 chains, 4000 iterations (2000 warm-up) and a thinning rate of 1. Listing B.2 shows the model output summary. A Correct Classification Rate (CCR) $= 0.83$ is calculated as the sum of the diagonal divided by the sum of all outcomes as shown in the contingency table B.1.

B.1.2 Density and Trace Plots

All chains in the trace plots (see Figure B.1) converge indicating that we had sufficient samples in our dataset.

Figure B.2 shows the posterior distributions for the model parameters and they are all unimodal (centre around a single value) suggesting that our chains have converged to the same posterior.

```
Output of model 'fit':

Computed from 8000 by 9193 log-likelihood matrix

         Estimate    SE
elpd_loo  -3837.5  62.0
p_loo        97.4   4.0
looic      7675.1 124.0
------
Monte Carlo SE of elpd_loo is 0.1.

Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)     9189  100.0%  890
 (0.5, 0.7]   (ok)          4    0.0%  706
   (0.7, 1]   (bad)         0    0.0%  <NA>
   (1, Inf)   (very bad)    0    0.0%  <NA>

All Pareto k estimates are ok (k < 0.7).
See help('pareto-k-diagnostic') for details.

Output of model 'fit_parent_filename':

Computed from 8000 by 9193 log-likelihood matrix

         Estimate    SE
elpd_loo  -4550.6  60.6
p_loo        52.2   2.5
looic      9101.2 121.1
------
Monte Carlo SE of elpd_loo is 0.1.

Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)     9192  100.0%  3058
 (0.5, 0.7]   (ok)          1    0.0%  966
   (0.7, 1]   (bad)         0    0.0%  <NA>
   (1, Inf)   (very bad)    0    0.0%  <NA>

All Pareto k estimates are ok (k < 0.7).
See help('pareto-k-diagnostic') for details.

Model comparisons:
                    elpd_diff se_diff
fit                    0.0       0.0
fit_parent_filename -713.1      33.2
```

Listing B.1: Output from the *loo* function comparing the presenter detection model fit using the clip filename (fit) or the parent filename (fit_parent_filename).

```
 Family: categorical
  Links: muPartial = logit; muPerfect = logit
Formula: RC ~ Lighting + MultipleObjects + StudentHeads + PMLevel + (1 | filename)
   Data: dt_original_New (Number of observations: 9193)
  Draws: 4 chains, each with iter = 4000; warmup = 2000; thin = 1;
         total post-warmup draws = 8000

Group-Level Effects:
~filename (Number of levels: 76)
                        Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(muPartial_Intercept)     2.24      0.45     1.52     3.27 1.00     1843     3049
sd(muPerfect_Intercept)     1.73      0.15     1.46     2.06 1.00      969     1916

Population-Level Effects:
                            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
muPartial_Intercept            -5.26      0.72    -6.88    -4.05 1.00     2173     3153
muPerfect_Intercept             2.31      0.24     1.84     2.77 1.03      456     1075
muPartial_LightingLow          -0.08      0.78    -1.62     1.49 1.00     1809     2584
muPartial_MultipleObjectsYes    4.52      0.54     3.52     5.64 1.00     3458     4028
muPartial_StudentHeadsYes       1.07      0.49     0.14     2.05 1.00     3862     4927
muPartial_PMLevelLow           -1.09      0.20    -1.48    -0.70 1.00     7596     6093
muPerfect_LightingLow          -0.82      0.30    -1.39    -0.24 1.01      735     1657
muPerfect_MultipleObjectsYes   -4.13      0.62    -5.42    -2.99 1.00     3743     4669
muPerfect_StudentHeadsYes       0.10      0.21    -0.30     0.52 1.00     1952     2923
muPerfect_PMLevelLow           -0.94      0.07    -1.07    -0.81 1.00     7638     6369

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

Listing B.2: Summary of presenter detection model fit.

Table B.1.: A contingency table showing the presenter detection model predictions on the x axis and the actual data on the y axis. The overall CCR $= 0.83$.

	Failure	Partial	Perfect	Total
Failure	1193	56	316	1565
row %	76.2%	3.6%	20.2%	17.0%
col %	51.7%	29.0%	4.7%	
Partial	42	85	12	139
row %	30.2%	61.2%	8.6%	1.5%
col %	1.8%	44.0%	0.2%	
Perfect	1072	52	6365	7489
row %	14.3%	0.7%	85.0%	81.5%
col %	46.5%	26.9%	95.1%	
Total	2307	193	6693	9193
	25.1%	2.1%	72.8%	

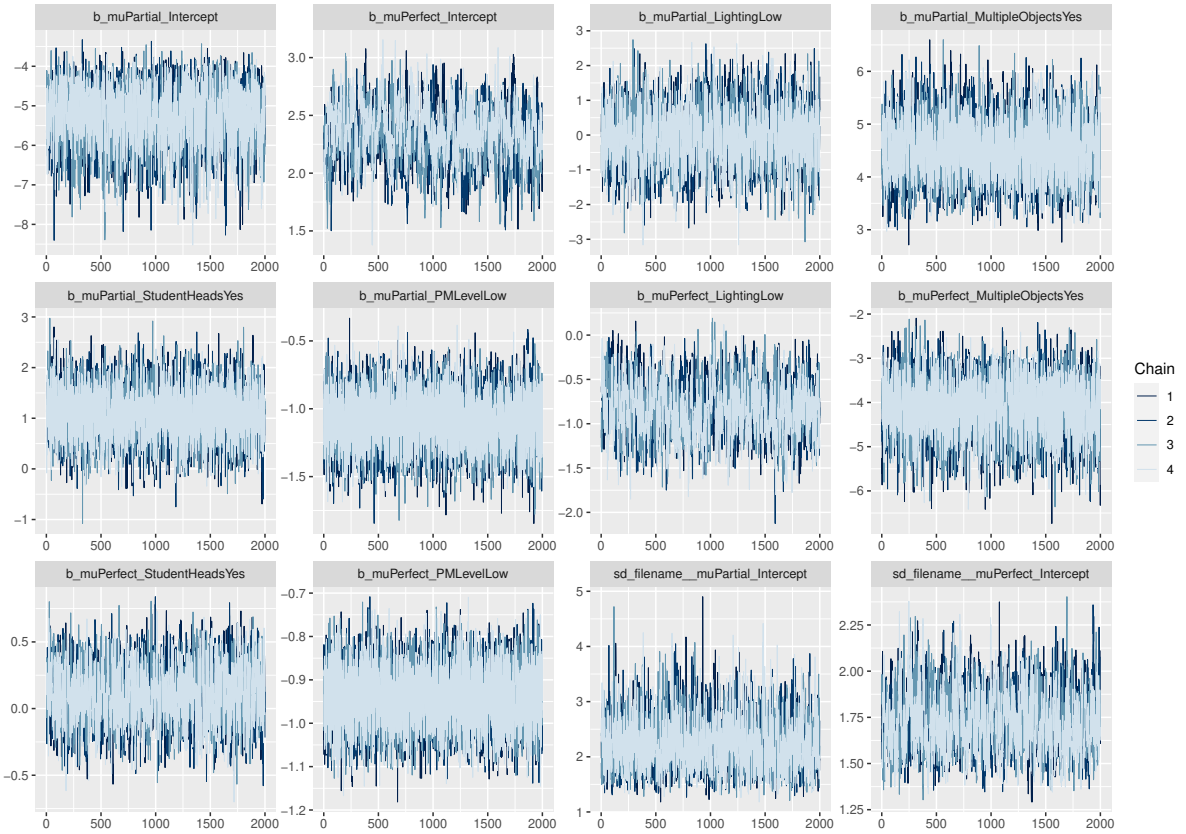


Figure B.1.: Trace plots showing the chains of all model parameters for presenter detection.

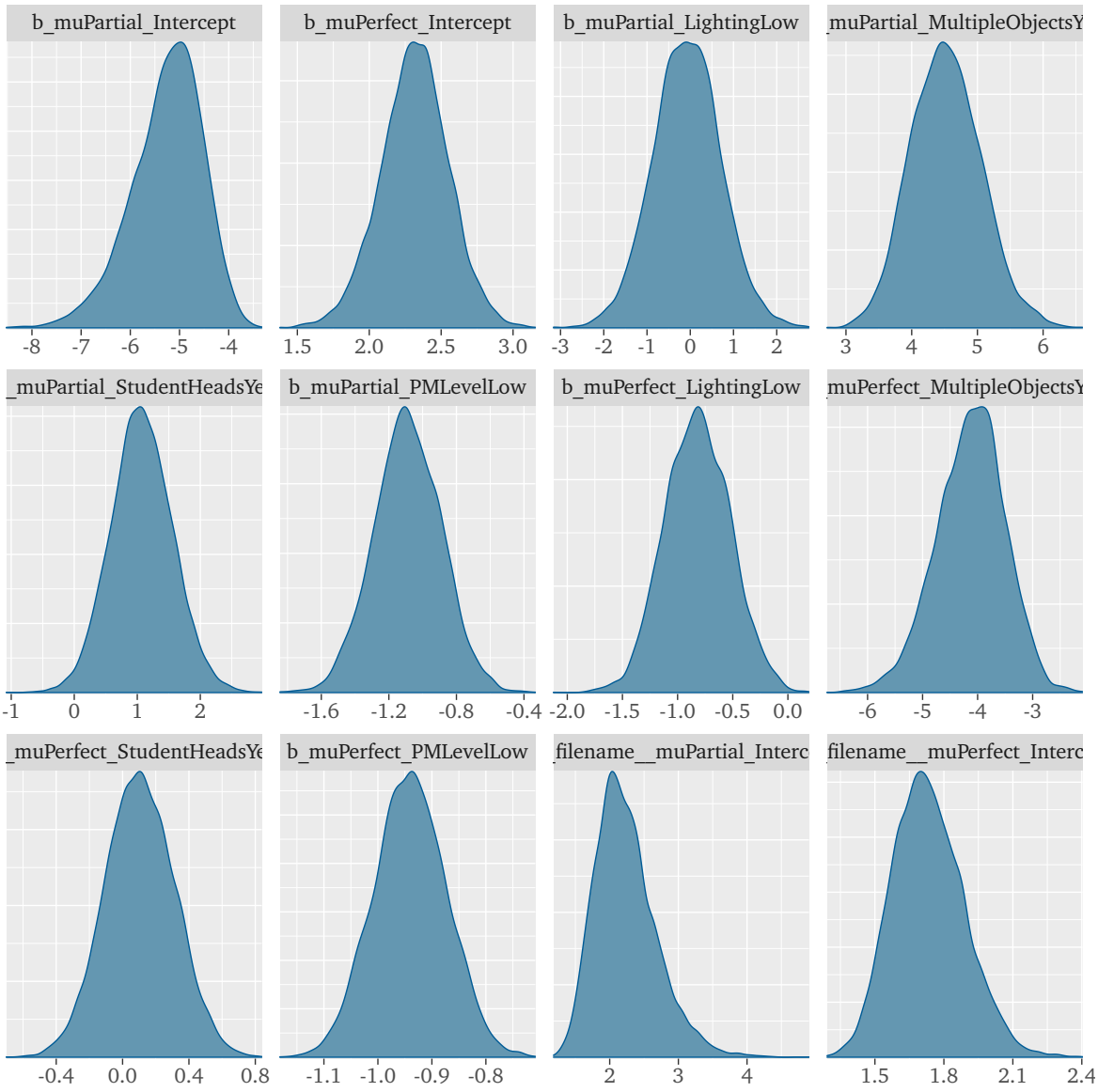


Figure B.2.: Densities of all model parameters for presenter detection.

Table B.2.: A contingency table showing the writing detection model predictions on the x axis and the actual data on the y axis. The overall CCR $= 0.73$.

	Failure	Partial	Perfect	Total
Failure	6852	715	633	8200
row %	83.6%	8.7%	7.7%	35.7%
col %	67.8%	9.7%	11.4%	
Partial	2091	6071	940	9102
row %	23.0%	66.7%	10.3%	39.6%
col %	20.7%	82.6%	17.0%	
Perfect	1161	566	3971	5698
row %	20.4%	9.9%	69.7%	24.8%
col %	11.5%	7.7%	71.6%	
Total	10104	7352	5544	23000
	43.9%	32.0%	24.1%	

B.1.3 Posterior Predictive Check

We used the pp_check function with 100 draws for each model and observed that all models produce synthetic data (y_{rep}) counts that closely mimic the original data (y) counts for each category shown in Figure B.3, indicating a good model fit.

B.2 Writing Detection

B.2.1 Model Parameters

Similar to the presenter detection modelling, using the clip filename as the random effect produced a better model fit (lower LOOIC value). The output from the *loo* function can be seen in Listing B.3.

Satisfactory model convergence was achieved ($\widehat{R} = 1$ for all parameters) using 4 chains, 2000 iterations (1000 warm-up) and a thinning rate of 1. Listing B.4 shows the model output summary. A Correct Classification Rate (CCR) $= 0.73$ was achieved and the prediction data is shown in the contingency table B.2.

B.2.2 Density and Trace Plots

All chains in the trace plots (see Figure B.4) converge indicating that we had sufficient samples in our dataset.

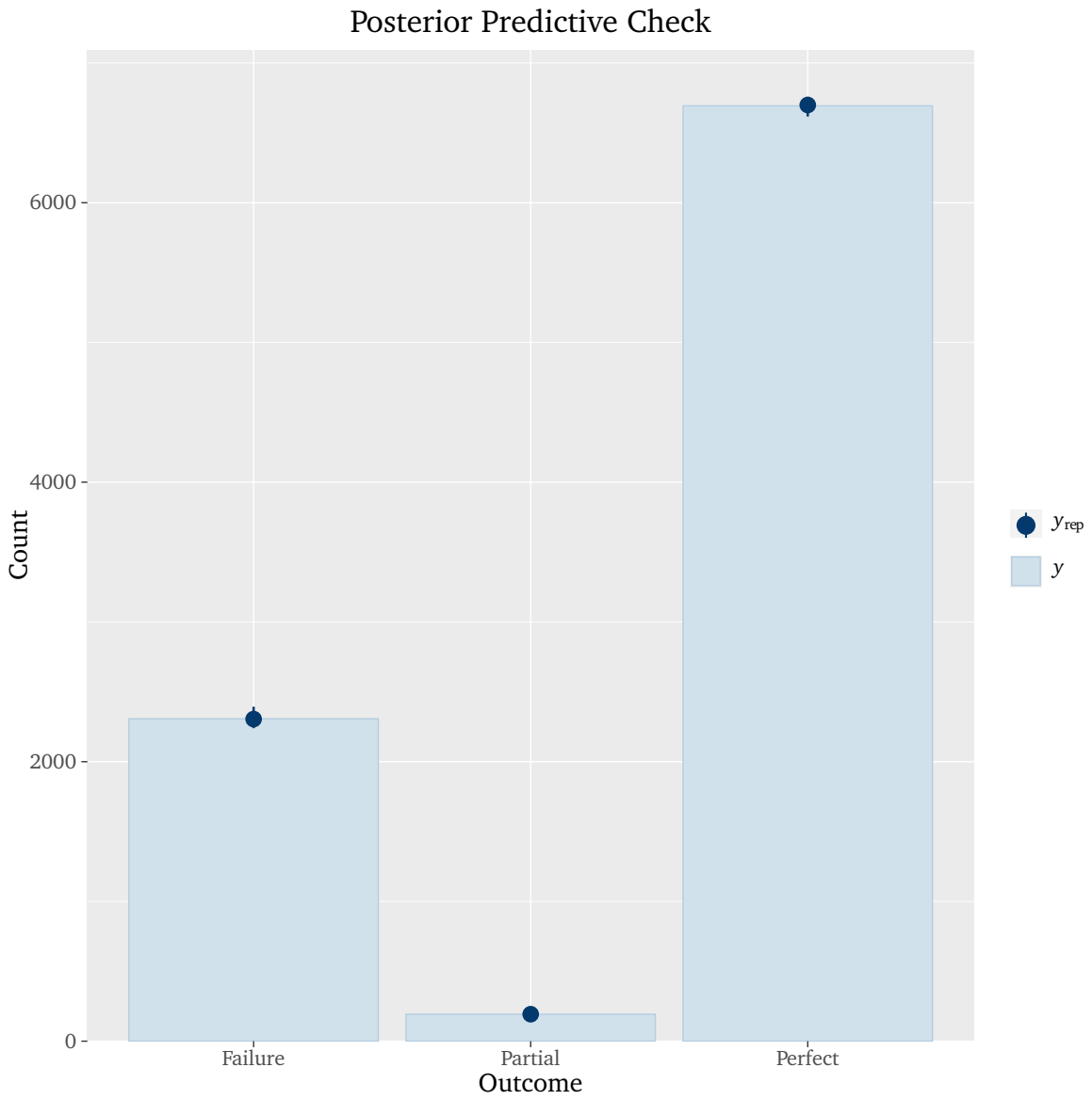


Figure B.3.: Posterior predictive check plot showing the actual data y and the synthetic draws y_{rep} for presenter detection.

```
Output of model 'fit':

Computed from 4000 by 23000 log-likelihood matrix

         Estimate    SE
elpd_loo -13920.2  92.9
p_loo        106.6   1.4
looic      27840.4 185.8
------
Monte Carlo SE of elpd_loo is 0.2.

Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)     22900 99.6%   988
 (0.5, 0.7]   (ok)         100  0.4%   3888
   (0.7, 1]   (bad)          0  0.0%   <NA>
   (1, Inf)   (very bad)     0  0.0%   <NA>

All Pareto k estimates are ok (k < 0.7).
See help('pareto-k-diagnostic') for details.

Output of model 'fit_parent_filename':

Computed from 4000 by 23000 log-likelihood matrix

         Estimate    SE
elpd_loo -17110.0  95.8
p_loo         61.0   0.9
looic      34220.1 191.7
------
Monte Carlo SE of elpd_loo is 0.1.

All Pareto k estimates are good (k < 0.5).
See help('pareto-k-diagnostic') for details.

Model comparisons:
                     elpd_diff se_diff
fit                      0.0       0.0
fit_parent_filename -3189.8       67.3
```

Listing B.3: Output from the *loo* function comparing the writing detection model fit using the clip filename (fit) or the parent filename (fit_parent_filename).

```
 Family: categorical
  Links: muPartial = logit; muPerfect = logit
Formula: RC ~ Lighting + ProjectorText + Diagrams + vis + (1 | filename)
   Data: dt_original_New (Number of observations: 23000)
  Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup draws = 4000

Group-Level Effects:
~filename (Number of levels: 76)
                      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
sd(muPartial_Intercept)     3.62      0.41     2.90     4.48 1.00      823     1718
sd(muPerfect_Intercept)     4.40      0.53     3.51     5.59 1.00     1088     1719

Population-Level Effects:
                          Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
muPartial_Intercept          -1.17      0.50    -2.20    -0.21 1.00      580     1201
muPerfect_Intercept          -3.10      0.68    -4.42    -1.77 1.01      604      801
muPartial_LightingLow        -4.87      0.83    -6.62    -3.34 1.00      899     1651
muPartial_ProjectorTextYes    2.64      0.21     2.22     3.06 1.00     3895     2672
muPartial_DiagramsYes         1.17      0.36     0.48     1.87 1.00     1412     1481
muPartial_visReduced         -1.13      0.16    -1.44    -0.82 1.00     2793     2770
muPerfect_LightingLow        -4.49      2.63   -10.21     0.13 1.00     2055     2193
muPerfect_ProjectorTextYes    4.97      2.62     0.45    10.51 1.00     2041     2328
muPerfect_DiagramsYes         0.05      0.20    -0.34     0.43 1.00     2825     2691
muPerfect_visReduced          0.21      0.17    -0.12     0.55 1.00     2942     2886

Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
and Tail_ESS are effective sample size measures, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

Listing B.4: Summary of writing detection model fit.

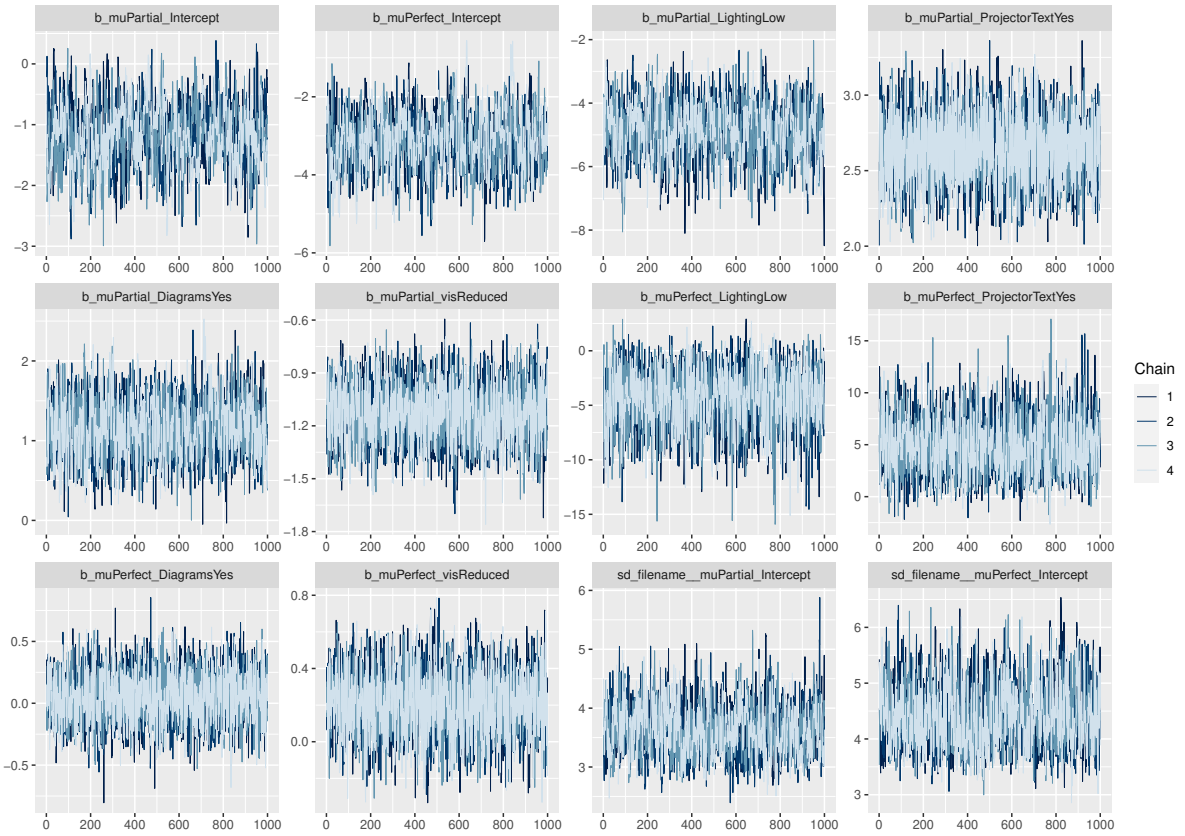


Figure B.4.: Trace plots showing the chains of all model parameters for writing detection.

Figure B.5 shows the posterior distributions for the model parameters and they are all unimodal (centre around a single value) suggesting that our chains have converged to the same posterior.

B.2.3 Posterior Predictive Check

We used the pp_check function with 100 draws for each model and observed that all models produce synthetic data (y_{rep}) counts that closely mimic the original data (y) counts for each category shown in Figure B.6, indicating a good model fit.

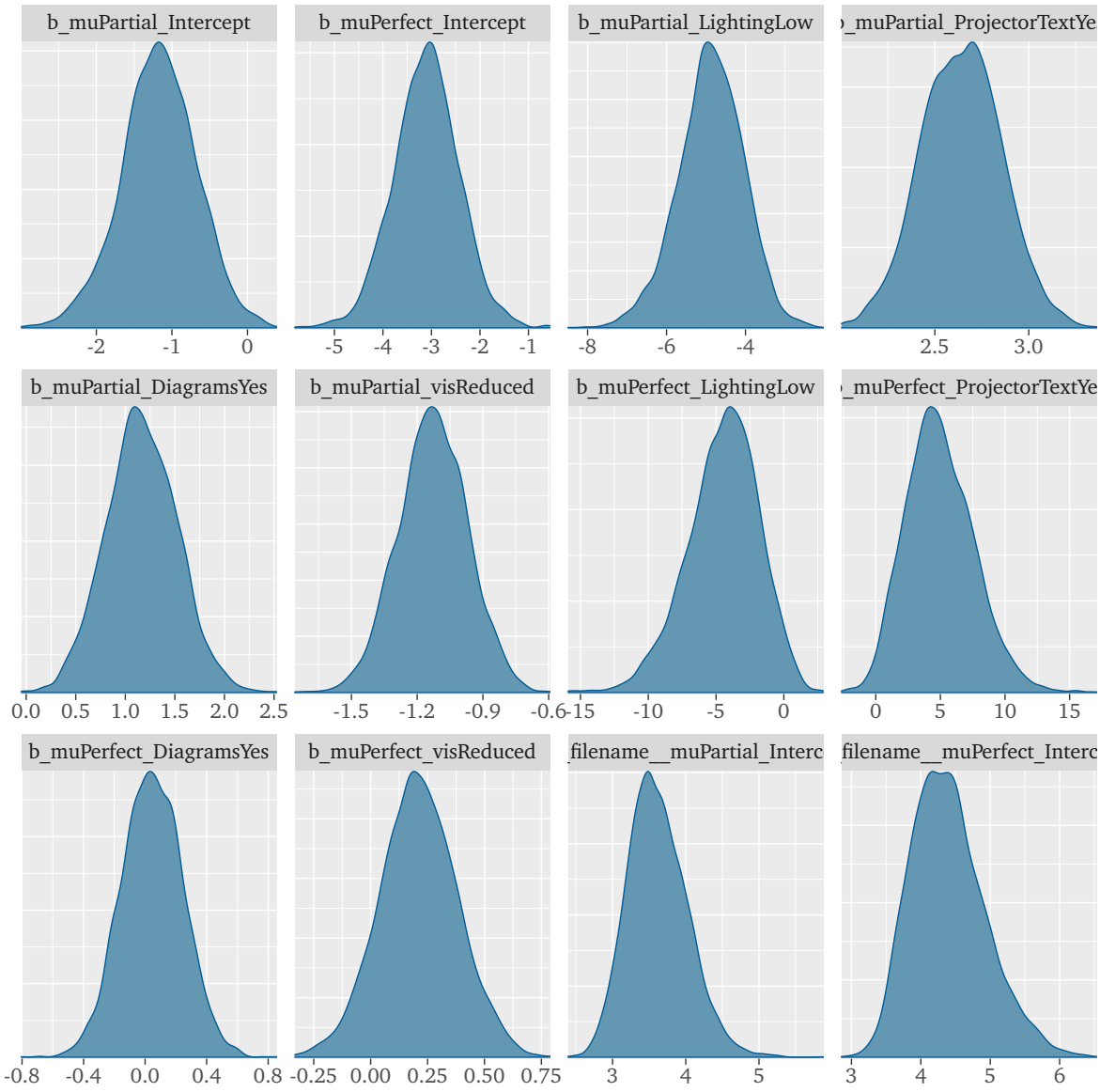


Figure B.5.: Densities of all model parameters for writing detection.

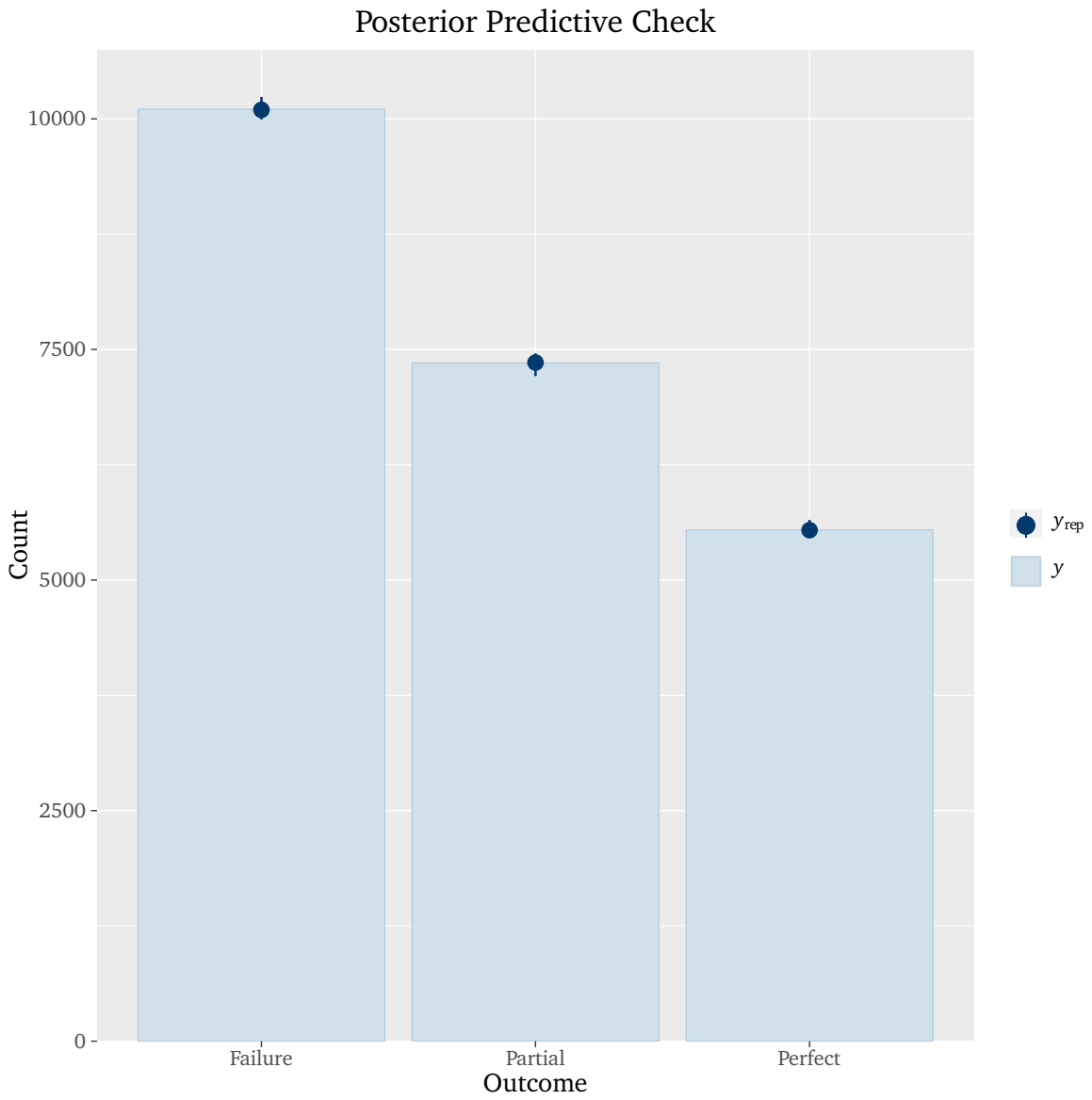


Figure B.6.: Posterior predictive check plot showing the actual data y and the synthetic draws y_{rep} for writing detection.

C

Writing Detection Performance Plots by Environmental Factor

The plots below show the overall performance graphs of the writing detection module under various environmental conditions. The plots were generated by filtering all frames in the testing dataset by the presence and absence of each factor (diagrams, low lighting, projector text and reduced visibility).

In each figure, (a) and (c) shows the plot with varying area recall constraint t_r while fixing $t_p = 0.4$ and (b) and (d) varying area precision constraint t_p while fixing $t_r = 0.8$. Each figure title indicates the condition represented and sub-plots (a) and (b) indicate the environmental factor being present while (c) and (d) indicates the absence of it.

The presence of diagrams (see Figure C.1) and reduced visibility (see Figure C.4) had the greatest impact on performance. In the presence of figures there is an almost linear decrease in precision as the recall constraint increases in the presence of figures (see Figure C.1a). In contrast, we observe a gradual decrease in precision and only declines sharply after about 80% of the recall constraint is reached (see Figure C.1c), in the absence of figures. A very similar observation is seen in during frames with reduced visibility as shown in Figure C.4.

The presence or absence of projector text or low lighting had very similar curves, suggesting that these factors had less of an impact on the detection accuracy when compared to the visibility of the board content and the presence of diagrams.

Figure C.1a and C.1b indicate a lower overall precision and recall in the presence of diagrams. Figure C.1c also shows a more gradual decline in accuracy as the area recall quality constraint approaches 1 and only begins to decrease rapidly after $t_r > 0.8$. In contrast, Figure C.1a shows a more constant decline that is almost linear, further indicating that in the presence of diagrams, the majority of detections are smaller than the GT. This is a limitation of the GT only providing a single tightest bounding box per board as discussed previously and shown in Figure 4.3b in Chapter 4.

Figure C.4c and C.4d indicate a higher overall precision and recall that is sustained for a larger range of quality constraints when there is no reduction in visibility as opposed to a reduction in visibility as shown in Figure C.4a and C.4b. The reduced visibility graphs show an almost linear

decrease that indicates that most of the detected rectangles are smaller than the GT rectangles. This is plausible since the GT annotation would enclose all text correctly regardless of reduced visibility.

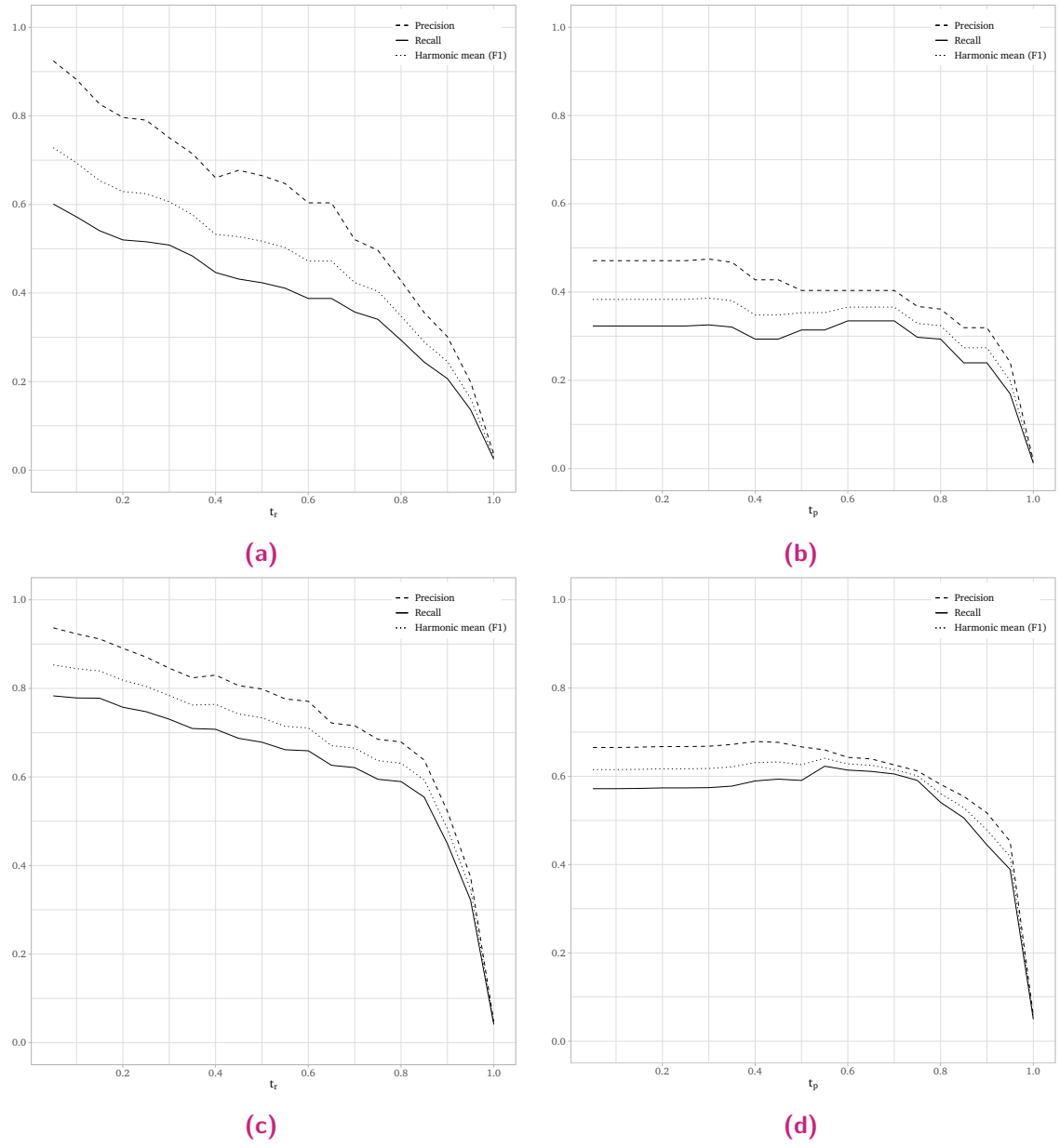


Figure C.1.: Performance plots for the presence (a and b) and absence (c and d) of diagrams on any boards.

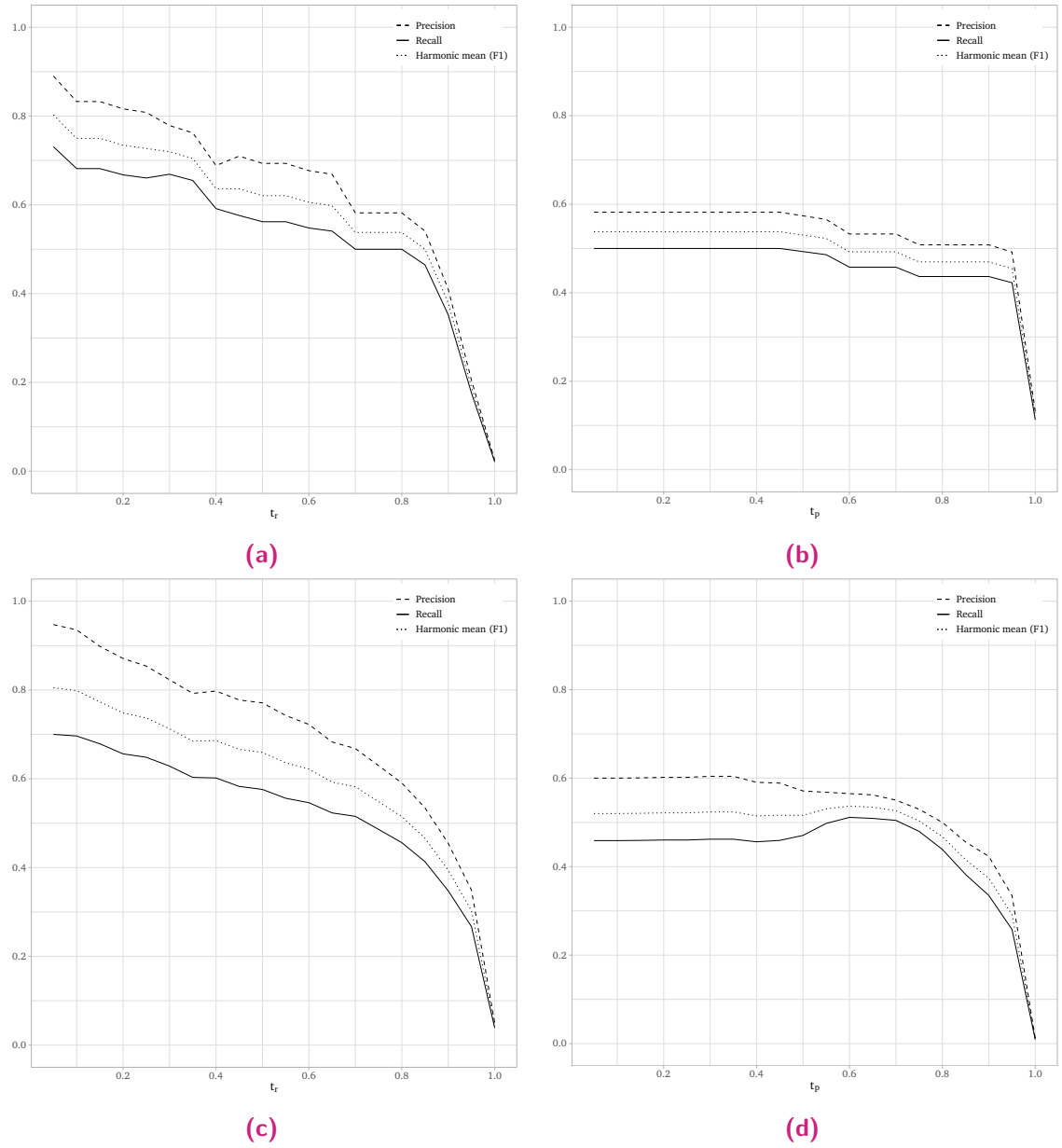


Figure C.2.: Performance plots for the presence (a and b) and absence (c and d) of low lighting.

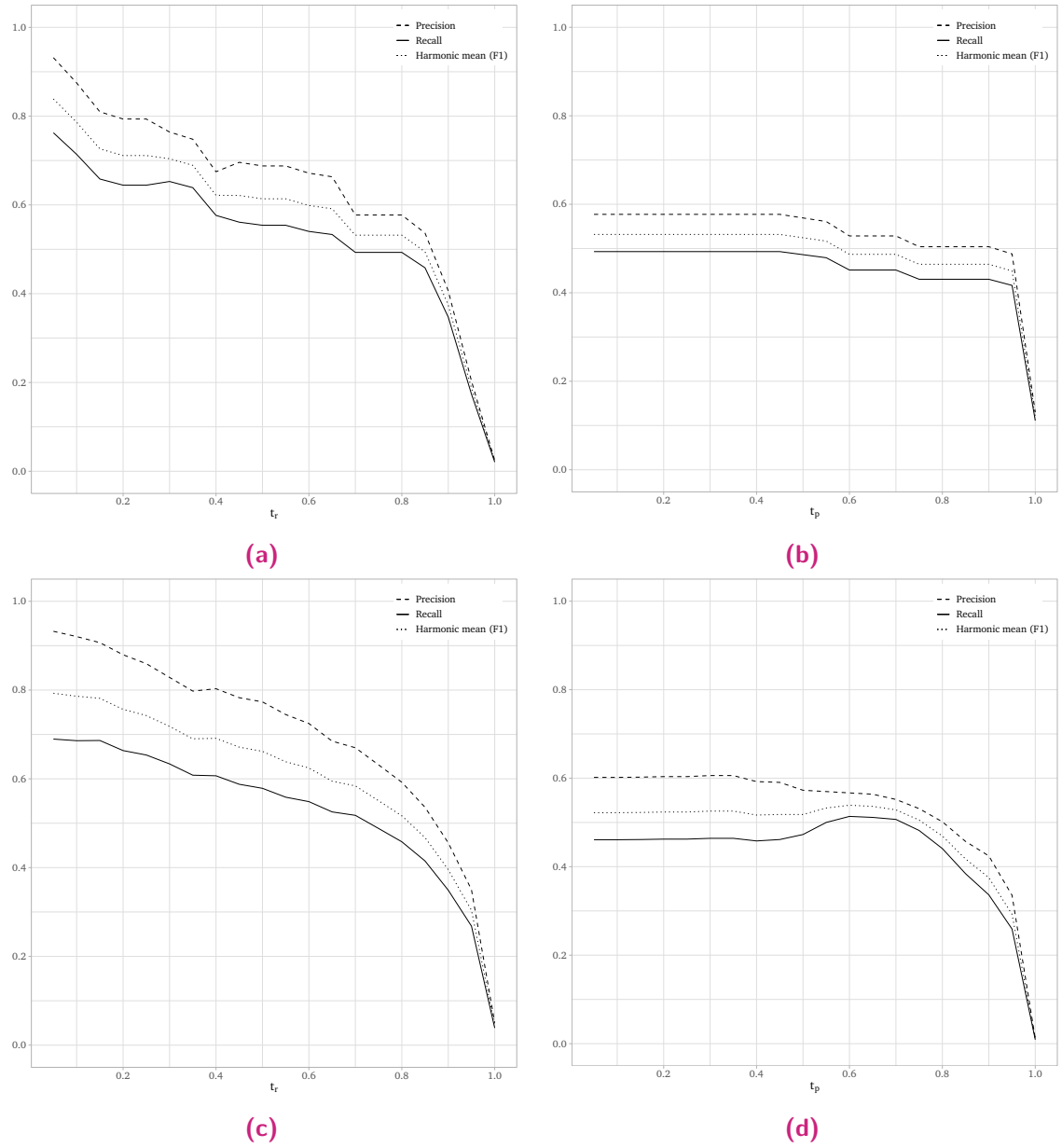


Figure C.3.: Performance plots for the presence (a and b) and absence (c and d) of projector text on any boards.

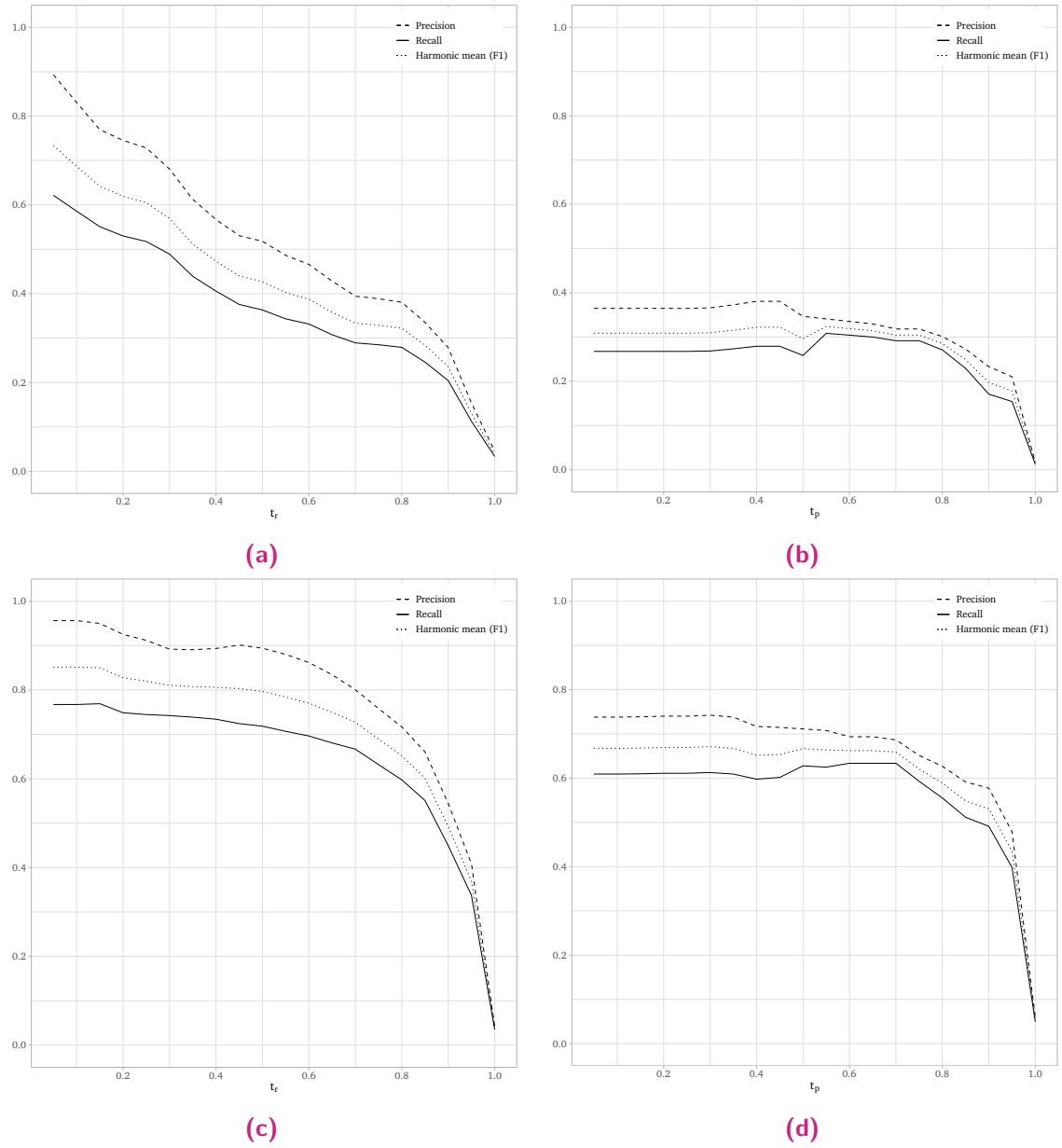


Figure C.4.: Performance plots for the presence (a and b) and absence (c and d) of reduced visibility on any boards.

Output File Formats

D

D.1 Configuration File

Below is the contents of the (*config.json*) file followed by a brief summary of each parameter and its purpose. These parameter values are also the default values that the program was compiled with and are used if no configuration file is specified as a command line argument. For detailed explanations of the parameters, refer to Chapter 3.

```
{
  "skipFrames": 0,
  "workingDimension": [ 3840, 2160 ],
  "maxFrames": 100,
  "intensityThresholdFindingImageMask": 15,
  "motionBoxOcclusionFactor": 0.3,
  "modelFilename": "./frozen_east_text_detection.pb",
  "writingDetectionSkipTime": 30,
  "writingConfThreshold": 0.5,
  "writingNmsThreshold": 0.4,
  "writingDetectionImgWidth": 0,
  "writingClusterMinSize": 12,
  "gestureThreshold": 0.3,
}
```

skipFrames: Number of frames to ignore when reading input frames from the video file.

workingDimension: The maximum resolution that the input video frames are resized down to only if the input video resolution is greater than this value.

maxFrames: The maximum number of frames to read into memory per iteration of the main processing loop.

intensityThresholdFindingImageMask: This parameter is a threshold for finding the black privacy regions set on the cameras and creates a bounding box around the image and excludes the black regions when the pixel intensity is less than this value.

motionBoxOcclusionFactor: This is the constant from Equation 3.2.

modelFilename: This is the file path for the pre-trained EAST network.

writingDetectionSkipTime: A time in seconds represents the amount of time to pass between writing detection calls.

writingConfThreshold: Only EAST predictions with a confidence greater than this value are considered.

writingNmsThreshold: Non-Maximum Suppression algorithm threshold used to remove overlapping predictions caused by EAST.

writingDetectionImgWidth: Width of the square image used by EAST. If set to 0, then the *workingDimension* width is used.

writingClusterMinSize: Minimum number of points belonging to a cluster for it to be considered. This translates to three words since each word bounding box has four vertices.

gestureThreshold: The percentage difference of white pixel distribution between the left and right half of the image searched for gestures.

D.2 Primary Output File

Below is a sample form of the main output file produced by our system followed by a short description of each field in the file.

```
{
"inputVideoFileName": "video_001.mp4",
"totalFrames": 706,
"fps": 29.7,
"skipFrames": 5,
"overallCrop": [ 0, 49, 1280, 626 ],
"workingDimension": [ 1280, 720 ],
"totalInputFrames": 3529,
"metaFrames": [
  {
  "frameNumberA": 0,
  "frameNumberB": 5,
```

```
  "timestampFrameA": 0.0,
  "timestampFrameB": 1.6823862598784672e+02,
  "multipresenter": 0,
  "containsBoardData": 1,
  "Presenters": [
    {
    "presenterIndex": 0,
    "presenterBounding": [ 830, 332, 108, 261 ],
    "deleted": 0
    }],
  "Clusters": [
    {
    "clusterBoundingBox": [ 935, 351, 116, 73 ],
    "numFeatures": 5,
    "rectsInGroup": [ 1.01161169e+03, 3.56722076e+02, 1.9414297
       1e+01, 1.03070021e+01]
    }
  ]
  },
  {
  "frameNumberA": 5, "frameNumberB": 10,
  "timestampFrameA": 1.6823862598784672e+02,
  "timestampFrameB": 3.3647725197569343e+02,
  "multipresenter": 0,
  "containsBoardData": 0,
  "Presenters": [
    {
    "presenterIndex": 0,
    "presenterBounding": [ 827, 336, 101, 257 ],
    "deleted": 1
    },
    {
    "presenterIndex": 28,
    "presenterBounding": [ 813, 345, 133, 221 ],
    "deleted": 0
    }
  ],
```

```
49    "Clusters": []
50    }
51    ]}
```

inputVideoFileName: File name of the input video this data belongs to.

totalFrames: The total number of *MetaFrame* records contained in this output file.

fps: The frame rate of the input video.

skipFrames: The *skipFrames*value that was used when processing this input file.

overallCrop: This is a rectangle bounding box which excludes the black region around any frame caused by a privacy region being set up on the physical camera.

workingDimension: The *workingDimension* value that was used when processing this input file.

totalInputFrames: The total number of frames in the input video file.

metaFrames: A list of serialised *MetaFrame* objects. Each *MetaFrame* object contains the following sub-objects:

- **Presenters:** A list of serialised *MovingEntity* objects. Each object contains the following fields:
  - **presenterIndex:** The index serves as a unique ID for each *MovingEntity* on a frame.
  - **presenterBounding:** The bounding box of the *MovingEntity* object.
  - **deleted:** A boolean value indicating whether this *MovingEntity* object was deleted and should be ignored by the VC. 0 = False and 1 = True.
- **Clusters:** A list of serialised *WritingGroup* objects. Each object contains the following fields:
  - **clusterBoundingBox:** A single bounding rectangle which is the boundary of all the word rectangles belonging to this cluster.
  - **numFeatures:** This is the number of word bounding boxes ideally bounding individual words.
  - **rectsInGroup:** This list contains the bounding box vertices for each of the word bounding boxes inside this cluster.

Dataset

E

Table E.1.: List of input videos used and the number of clips extracted from each series of full-length videos.

Lecture Series ID	Number of Full Length Videos	Number of 2-minute clips
1	6	12
2	7	14
3	1	2
4	2	14
5	1	4
6	3	12
7	1	11
8	3	23
9	1	8

Table E.2.: This table shows the ground truth dataset with all the input videos categorised into either the testing set (group 0) or training set (group 1). It also shows the amount of each attribute present in each particular video file. The red shades of colour for the video length column indicate videos that deviate from the 120 second length. The colour coding for the rest of the table is a scale from blue (0) to red representing the cell value.

Group	Video ID	Video Length (Seconds)	Heads	Presenter	Other	Movement	Gesture	Referencing Gesture	Handwritten text	Projector text	Busy Scene	Low lighting	Active boards	Poor visibility	Poor contrast	Shadows	Characters	Diagrams	Dusty boards
0	1	121.09	2121	3629	1036	4.70	0	0	0	7258	1026	3629	952	0	0	0	0	0	0
0	2	120.35	4232	3607	1110	3.88	0	0	0	7214	872	3607	0	0	0	0	0	0	0
1	3	120.62	4213	1188	0	3.01	61	77	0	7230	0	3615	0	0	0	0	0	0	0
1	4	120.79	3354	3620	1831	3.66	0	0	0	121	1857	3620	121	121	0	0	0	121	0
1	5	119.85	405	3592	0	4.52	13	1025	0	7184	0	3592	6936	0	0	0	0	5170	0
0	6	120.55	57	3613	109	6.74	226	0	0	7226	159	3613	0	0	0	0	0	0	0
0	7	120.59	576	3614	2551	2.28	0	0	0	7212	1000	3614	6626	2794	0	0	0	0	0
0	8	120.42	4149	3609	0	3.86	167	14	0	7218	0	3609	2176	2164	0	0	0	2176	0
0	9	119.69	0	3585	0	1.92	0	0	0	7174	324	3587	0	7174	0	0	7174	7174	0
0	10	120.32	0	3605	0	5.67	0	0	0	7212	0	3606	0	0	0	0	0	7212	0
0	11	120.82	0	3618	0	4.09	0	0	0	7242	0	3621	0	0	0	0	0	0	0
1	12	120.25	0	1565	0	3.51	0	0	0	7208	0	3604	0	0	0	0	0	0	0
0	13	121.39	7348	3638	0	4.65	280	836	0	7276	0	3638	7276	6834	0	0	6834	6834	0
1	14	120.99	5429	3626	0	3.96	292	1137	0	7252	0	3626	7252	3156	0	0	3156	3438	0
0	15	119.49	7116	3581	0	5.46	0	2128	0	7162	0	3581	7162	3598	0	0	0	7162	0
0	16	121.09	5734	3629	0	4.35	295	857	0	7258	0	3629	7258	0	0	0	0	3246	0
0	17	120.92	5160	3624	0	5.00	188	1024	0	7248	0	3624	7248	3880	0	0	0	7156	0
0	18	120.25	0	3573	0	5.65	0	0	0	0	3604	900	0	0	0	0	0	0	0
0	19	120.40	6450	1806	0	6.99	0	1032	0	3612	0	1806	3612	2092	0	0	0	2674	0
0	20	120.45	1768	3610	28	2.38	0	0	7220	0	0	3610	0	7220	7220	0	7220	3610	0
0	21	120.29	6	3455	0	3.87	86	0	1581	0	0	3605	0	1581	1581	0	1581	856	856
0	22	121.66	3634	3646	0	3.09	46	341	0	0	0	3646	0	0	0	0	0	0	0
0	23	120.69	969	3617	0	5.36	227	508	0	6432	0	3617	6432	0	0	0	0	1387	0
0	24	120.52	848	3558	0	3.60	70	157	0	7224	0	3612	7224	0	0	0	0	0	0
0	25	120.29	402	3476	0	4.45	121	504	0	7210	0	3605	7210	7210	7210	0	0	7210	0
1	26	121.32	3579	3636	0	5.34	35	901	0	7272	0	3636	7272	0	0	0	6906	6906	0
0	27	131.70	50	3947	0	5.06	0	0	3947	7894	0	0	3597	11841	3947	0	11841	7894	3947
1	28	125.03	0	3747	0	5.19	0	86	5578	7494	0	0	5393	5153	5578	0	13072	11411	5578
0	29	119.55	0	3169	0	4.25	0	0	0	0	1821	0	0	0	0	0	0	0	0
1	30	120.42	7	3609	0	3.89	0	613	3195	0	0	0	3195	0	0	0	3025	0	0
0	31	120.19	3282	3602	0	3.72	0	433	6324	0	0	0	3442	0	0	0	6164	0	0
0	32	120.99	3294	3626	0	4.14	0	117	9768	0	0	0	3455	0	0	0	9458	3626	0
0	33	119.89	3365	3593	0	3.82	0	0	11463	0	0	0	3593	0	1258	0	11203	3593	0
0	34	120.75	2799	3619	0	3.06	13	508	17805	0	0	0	3329	0	0	0	17805	9167	0

1	35	121.56	2311	3313	0	5.59	0	73	17518	0	0	0	2294	0	0	0	13095	7746	1518
0	36	121.46	3305	3640	0	3.38	0	43	21500	0	0	0	3640	0	0	0	21500	10580	6940
1	37	119.72	3190	6928	65	2.08	37	48	6576	0	3340	0	746	3934	6576	0	6576	0	6576
0	38	121.16	2227	7262	0	3.41	0	0	0	0	3631	0	0	0	0	0	0	0	0
0	39	114.72	3164	3438	0	4.04	0	54	1321	0	0	0	1321	0	0	0	391	0	1321
0	40	120.65	4664	3616	0	4.47	0	356	3616	0	0	0	3436	0	0	0	3616	3286	3616
0	41	120.15	3180	3601	0	2.58	0	946	21606	0	0	0	0	0	0	0	21606	10803	21606
1	42	120.12	1032	3599	0	5.99	0	0	21600	0	2685	0	0	0	0	0	21600	10800	21600
0	43	120.05	0	3598	0	2.58	0	0	7196	2682	0	3598	2682	0	0	0	7196	7196	0
0	44	120.82	0	3621	0	2.65	0	0	7242	3621	0	3621	0	7242	7242	0	7242	7242	0
0	45	120.12	0	3600	0	2.97	32	0	7200	3600	0	3600	3600	7543	7200	0	7200	7200	0
0	46	121.29	0	2849	0	4.20	0	0	7270	2117	0	3635	2117	7794	7301	0	7270	7270	0
0	47	119.62	3713	2077	0	3.35	0	0	7170	0	0	0	0	7170	7170	0	7170	0	0
1	48	121.32	6792	3636	0	4.76	105	10	7272	0	0	0	0	7272	7272	0	7272	0	0
1	49	121.06	2497	3628	0	5.60	266	62	7256	0	0	0	0	7256	7256	0	7256	0	0
1	50	120.86	26	3622	0	4.06	0	0	7244	0	0	0	0	7244	7244	0	7244	0	0
0	51	120.99	0	3626	0	3.88	0	0	7252	0	3566	0	0	7252	7252	0	7252	0	0
0	52	119.46	10365	3564	0	5.59	0	0	14256	0	0	430	0	1720	1720	0	14256	10692	0
0	53	120.75	10003	3619	0	2.46	0	0	6408	992	0	1589	992	992	582	270	6408	4566	0
0	54	121.06	8727	3628	0	1.52	0	0	0	7256	0	3628	7256	7256	0	0	0	0	0
1	55	119.89	9050	3593	0	4.53	0	0	7932	782	0	3593	782	6916	7932	0	7932	5834	0
0	56	120.52	10283	3612	2	6.20	153	12	14448	0	0	0	0	14448	14448	0	14448	10836	0
0	57	120.99	7511	3626	0	4.36	150	17	14504	0	0	0	0	14504	14504	0	14504	10878	0
1	58	121.46	0	3639	0	3.62	0	0	4738	0	0	3640	0	4738	4738	0	4738	4738	0
1	59	120.70	0	3592	0	5.85	0	80	6702	0	0	0	3081	0	0	0	6023	0	3081
0	60	120.99	0	3582	0	8.38	32	303	7252	0	0	0	3546	0	0	0	7252	0	3626
0	61	120.39	0	3608	0	7.04	429	148	10244	0	0	0	3608	0	0	0	6636	0	3608
0	62	120.22	0	3603	0	6.73	0	689	10534	0	0	0	3603	0	0	1722	9878	0	8202
1	63	120.13	0	3604	0	7.21	210	547	10812	0	0	0	3604	0	0	7208	10812	0	10812
0	64	119.72	0	3428	0	6.61	0	0	13233	0	0	0	2519	0	0	3757	10714	0	7345
0	65	120.92	0	3609	0	5.87	0	636	10872	0	0	0	3624	0	0	3624	10872	0	7248
0	66	120.17	0	3199	0	8.70	0	116	10395	0	0	0	2955	0	0	60	10395	0	6810
1	67	120.44	0	3410	0	8.09	0	143	10809	0	0	0	3353	0	0	0	10809	0	7206
0	68	120.25	0	3604	0	4.93	324	123	14416	0	0	0	0	0	0	0	14416	0	10812
0	69	120.29	0	2835	0	4.38	0	0	1853	0	0	0	1853	0	0	0	0	0	0
0	70	122.92	0	3684	0	5.23	0	0	3684	0	0	0	0	0	0	0	3684	0	3684
0	71	120.13	0	3594	0	5.29	0	1691	10782	0	0	0	2933	0	0	0	10782	0	7188
1	72	110.43	0	3146	0	7.03	17	695	13252	0	0	0	0	0	0	0	9939	3313	6626
0	73	120.19	0	3602	0	6.37	52	206	14408	0	3527	0	0	0	0	0	10806	3602	7204
0	74	120.34	0	2969	0	5.03	0	0	1164	0	0	0	0	0	0	0	0	0	0
0	75	120.77	0	3549	0	6.73	3	676	3623	0	0	0	3623	0	0	0	3623	0	0
1	76	120.73	0	3577	0	6.17	0	535	7244	0	0	0	3622	0	0	0	6944	0	0
0	77	119.65	0	3103	0	4.73	0	789	7172	0	0	0	3586	0	0	0	7172	0	0
0	78	121.02	0	3052	0	4.68	66	1436	7254	0	0	0	1667	0	0	0	7254	0	0
0	79	121.07	0	3632	0	3.57	170	297	10896	0	0	0	0	0	0	3632	10896	3632	0
0	80	119.75	0	3589	0	5.72	25	249	14356	0	0	0	3136	0	0	7178	14356	3589	0
0	81	120.89	0	3623	0	6.95	0	457	14492	0	0	0	3493	0	0	0	14492	0	3623

0	82	120.59	0	3289	0	7.09	25	0	0	0	0	0	0	0	0	0	0	0	0
0	83	120.59	0	3614	0	2.28	0	0	0	0	0	0	0	0	0	0	0	0	0
0	84	120.00	0	3600	0	5.16	0	417	3270	0	0	0	0	0	3270	0	981	0	3270
0	85	121.36	0	3637	0	5.14	0	1109	7274	0	0	0	3467	3637	7274	0	7274	0	7274
0	86	120.65	0	3616	0	6.38	0	742	7232	0	0	0	2358	3616	7157	0	7232	0	7232
0	87	120.75	0	3619	0	6.45	0	44	14476	0	0	0	3619	0	14476	0	14476	0	14476
0	88	120.55	0	3613	0	5.47	208	420	14365	0	0	0	2636	0	14365	0	14365	0	14365
0	89	121.19	0	3632	0	4.40	0	0	14528	0	2979	0	0	0	14528	0	14528	0	14528
0	90	120.29	0	3605	0	4.97	184	141	7212	0	522	0	0	0	0	0	7212	0	0
1	91	120.92	0	3594	0	5.39	117	571	7188	0	0	0	2210	0	0	0	7188	0	0
0	92	121.06	0	3628	0	4.83	44	622	8330	0	0	0	3628	0	0	0	7780	0	0
1	93	120.06	0	3598	0	4.13	0	185	14753	0	0	0	3598	0	0	0	14753	0	0
0	94	115.15	0	3451	0	5.03	0	243	17255	0	0	0	3331	0	0	0	17255	0	0
0	95	120.42	0	3609	0	4.87	0	124	20124	0	0	0	3539	0	0	0	19492	0	0
0	96	121.05	0	3628	0	4.65	0	236	20837	0	0	0	2697	0	0	0	20199	0	0
1	97	118.74	0	3529	0	5.41	0	571	18077	0	0	0	3286	474	0	474	17566	0	2875
0	98	121.03	0	3627	0	4.41	31	13	18135	0	0	0	3567	3627	0	3627	18135	0	10881
0	99	120.19	0	3602	0	5.50	0	575	18575	0	0	0	2239	12	0	1089	18575	0	18575
0	100	119.86	0	3591	25	3.83	0	658	17960	0	1928	0	3592	0	0	0	17960	0	17960

Table E.3.: This table shows the similarity between the test and training datasets. The training and test set rows show the average count for each object and attribute. The similarity row indicates the similarity between the counts as a fraction between 0 and 1, where 1 indicates a perfect split between the two groups for that particular object or attribute.

Group	Video Length (Seconds)	Heads	Presenter	Other	Movement	Gesture	Referencing Gesture	Handwritten text	Projector text	Busy Scene	Low lighting	Active boards	Poor visibility	Poor contrast	Shadows	Characters	Diagrams	Dusty boards
Training set	120.56	1717.13	3550.68	63.96	4.70	47.99	303.25	7306.58	1833.42	328.41	1156.05	2374.41	1884.22	1969.80	328.41	7342.51	2399.33	2844.70
Test set	120.31	1745.21	3524.63	79.00	4.94	48.04	306.63	7406.08	1855.96	328.42	1205.25	2364.38	1927.67	1941.50	320.08	7746.08	2478.21	2744.67
Similarity	1.00	0.98	0.99	0.81	0.95	1.00	0.99	0.99	0.99	1.00	0.96	1.00	0.98	0.99	0.97	0.95	0.97	0.96