5-14-2021

Lecture Video Transformation through An Intelligent Analysis and Post-processing System

Xi Wang
University of Massachusetts Amherst

Lecture Video Transformation through An Intelligent Analysis and Post-processing System

A Thesis Presented

by

XI WANG

Submitted to the Graduate School of
the University of Massachusetts Amherst
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

May 2021

College of Engineering, Electrical & Computer Engineering

Lecture Video Transformation through An Intelligent Analysis and Post-processing System

A Thesis Presented

By

XI WANG

Approved as to style and content by:

_____

Professor Lixin Gao, Chair

_____

Professor Russell Tessier, Member

_____

Professor Michael Zink, Member

_____

Christopher V. Hollot, Department Head

Electrical and Computer Engineering

DEDICATION

To my committee.

ACKNOWLEDGEMENTS

Thanks to my advisor, Professor G, for her thoughtful and patient guidance. I would also like to extend my gratitude to the fantastic committee members, Professor R and Professor Z, for their comments and suggestions on the stages in this project.

I want to thank Multimedia Networking & Internet Lab, which Professor G supervises, for allowing me to use NVIDIA ® Quadro ® 5000 GPU.

I wish to express my appreciation to all the individuals who volunteered their participation in this project.

A special thank you to all those whose support and friendship helped me to stay focused on this project and who provided me with the encouragement to continue when things got difficult.

ABSTRACT

LECTURE VIDEO TRANSFORMATION THROUGH AN INTELLEGENT ANALYSIS AND POST-PROCESSING SYSTEM

MAY 2021

XI WANG

B.A., BEIJING CITY UNIVERSITY

M.S.E.C.E., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Lixin Gao

Lecture videos are good sources for people to learn new things. Students commonly use online videos to explore various domains. However, some recorded videos are posted on online platforms without being post-processed due to technology and resource limitations. In this work, we focus on the research of developing an intelligent system to automatically extract essential information, including the main instructor and screen, in a lecture video in several scenarios by using modern deep learning techniques. This thesis aims to combine the extracted essential information to render the videos and generate a new layout with smaller file size than the original one. Another benefit of using this approach is that the users may save video post-processing time and costs. State-of-the-art object detection models, an algorithm to correct screen display, tracking the instructor, and other deep learning techniques were adopted in the system to detect both the main instructor and the screen in given videos without much of the computational burden.

There are four main contributions:

1. We built an intelligent video analysis and post-processing system to extract and reframe detected objects from lecture videos.

2. We proposed a post-processing algorithm to localize the frontal human torso position in processing a sequence of frames in the videos.

3. We proposed a novel deep learning approach to distinguish the main instructor from other instructors or audience in several complex situations.

4. We proposed an algorithm to extract the four edge points of a screen at the pixel level and correct the screen display in various scenarios.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Videos, which are good sources for exploring new things and could be stored as history files, have become ubiquitous and can be easily accessed on the Internet. In recent years, the fast growth of video platforms and vast network bandwidth has allowed more people to study online and watch live and recorded lecture videos. As far as we know, due to different shooting conditions and technology limitations, some lecture videos are directly uploaded to the Internet without being post-processed. A better condition is that videos are reprocessed to generate higher quality ones by consuming extra time and labor resources.

We found the main instructor, together with many audiences, is presented in many lecture videos. Distracting factors, such as students and skewed screens degrade the users' viewing experience, making it challenging to focus on the main instructor and the main screen in a lecture. This work proposed and built an intelligent video reframing system that captures the main instructor and the main screen from original lecture videos. Then, these features are reframed in the system to render a new high-quality video, highlighting the main instructor and extracted screen. Building this system requires many essential techniques, including object detection, image processing, video processing, screen refinement, determining and tracking the main instructor, etc. In this work, an intelligent analysis and post-processing system was developed to automatically detect, extract, and reframe the main instructor and screen in videos recorded during a lecture or similar scenario.

Traditional approaches for reframing lecture videos to different aspect ratios usually involve complicated shooting settings and video post-processing methods [24]. For instance, a professional video curator controls a camera to track the main instructor during shooting. After shooting, the video curator needs to identify the main instructor in each frame, track the main instructor's physical location transitions from frame-to-frame, adjust crop target regions in the video, and determine what time the instructor was teaching which slide. After the curators recognize the main instructor and screen, corresponding frames are rendered to generate a new video containing the main instructor and screen. This process is tedious, time-consuming, and error-prone.

To address the above problems, we designed a novel intelligent video analysis and post-processing system to recognize, extract and reorganize target objects in lecture videos. In our deep learning object detection model, the main instructor and screen are typically detected while filtering out other noisy objects in various scenarios. At the video analysis stage, the system distinguishes the human's frontal view. Then, the system identifies the main instructor based on his/her displayed time in the scenarios of multiple persons or other ambiguous contexts (Sec. 3.3). Finally, the video analyzer outputs the bounding boxes of the main instructor and screen that were detected in the first frame. At the video rendering stage, this system initializes a tracker to track the main instructor. The screen looks skewed in some videos due to the shooting angles. For screen correction, the system first finds the screen contour using image processing algorithms and then uses the perspective transformation algorithm to correct the skewed screen. The main instructor tracking, and screen viewpoint correction occur in every frame. Then the system sets the video resolution depending on the corrected screen size. The system uses seamless padding to make the

video layout looks more comfortable than the original one does. As the experiments show (Figure. 1), the video quality is enhanced. Meanwhile, the video file size is reduced by an average of 50%.

Figure 1: New layout videos generated by the intelligent analysis and post-processing system. Above: original videos (input). Below: processed videos (output).

There are four main contributions in this work:

• This work built an intelligent video analysis and post-processing system that can detect, extract, and reorganize target objects, the main instructor, and the aspect screen from original lecture videos. It then converts the original video to a new layout. This system costs less time than traditional methods, such as video shooting and post-processing. At the same time, the video file size is reduced by 50% on average.

• This work proposes a post-processing method to identify the frontal human torso. We trained a detector to detect faces and human torsos. We also calculated the Intersection over Union (IoU) ratio, which is the detected face area over its corresponding detected torso's area, to evaluate which person presents his frontal torso in the video.

- This work demonstrates a new approach to combining a histogram-based classifier and time cues to find the main instructor's region of interest (ROI) to make sure the frames containing the main instructor are extracted in some ambiguous scenarios.

- This work proposes a contour extracting algorithm to find the rectangular outline of the screen. Once the most similar shape of the screen is found, the perspective transformation algorithm is supposed to be implemented at the pixel level. This algorithm could be extended to find the maximum area of irregular quadrilateral contours.

The structure of this thesis is shown in the following sections. In Chapter 2, we discuss related work. In Chapter 3, we describe the intelligent video analysis and post-processing system design, the reason for selecting the adopted object detection model, the approach to identify the frontal human torso, the main instructor identification, and screen correction techniques. Chapter 4 presents the experiment's results. Chapter 5 describes conclusions.

CHAPTER 2

RELATED WORK

In this chapter, we discuss previous studies related to frontal human torso detection and main instructor identification.

2.1 Frontal Human Torso Detection

The frontal human torso detection is a specific task in the human torso orientation estimation field. Traditional methods for human torso orientation estimation use an external camera or sensors. Bo Peng *et al.* [1] proposed a method to estimate human torso orientation using the skeleton gained from motion capture torso orientation. Angelo *et al.* [2] used magnetic sensors to estimate human torso orientation. Cheng *et al.* [3] combined the head and torso cues to estimate human torso orientation in surveillance videos. Kai-Chi *et al.* [4] used a 3D-Point-Cloud feature to assess the human pose. K. Yoo [5] proposed a frontal human torso detection method by using an object's real-measurement using a depth camera. Although these works show positive results, fixed cameras or sensors are needed, and feature extraction processes are complicated. Recently, researchers [6] [7] tried to use a deep neural network (DNN) to solve this problem. Byungtae Ahn *et al.* [6] designed a DNN architecture for estimating head orientation. Jinyoung *et al.* [7] proposed a lightweight classification convolutional neural network (CNN) based end-to-end system for estimating human torso orientation. In contrast to these previous studies, we propose a method to localize the frontal human torso position in a video sequence using the Interaction over Union (IoU) ratio by combining the detected face and torso cues.

2.2 Main Instructor Identification

Instructor detection is essential to determine the main instructor in the lecture videos. Recently, there have been many works on instructor identification in TV shows and movies. M. Everingham *et al.* and J. Sivic *et al.* [8, 9] use a face detector combined with the Gaussian mixture model (GMM) to determine the current speaking character detecting lip movements. Tapaswi *et al.* [10] train the GMMs using Expectation Maximization and use maximum a-posteriori probability to identify the instructor. More recently, Nagrani *et al.* [11] use the VoxCeleb datasets [12] to extract CNN-based feature vectors and train an SVM classifier to identify the instructor. These methods [8, 9, 10, 11, 12] require trained models to determine the instructor in TV series or movies. However, the main instructor in the lecture video shot is more evident than TV series or movies. Besides, existing state-of-the-art algorithms have excellent face verification with more computing resources.

In this work, we propose an efficient approach to determine the main instructor in the lecture videos. Our method is more portable and convenient because the proposed system analyzes the main instructor by comparing video streams frame by frame. Besides, less memory and fewer computing resources are required.

CHAPTER 3

METHODOLOGY

This chapter aims to present how to process a raw video with no captions or annotations to detect, extract, and combine the main instructor and screen. After the detection, the proposed system combines these extracted components to generate a new layout video. We organize this section as follows. In Section 3.1, we introduce the design of the proposed video analysis and post-processing system. In Section 3.2, we describe the video analyzer architecture in the system. In Section 3.3, we present the architecture of the video renderer.

3.1 The video analysis and post-processing system overview

In this part, we introduce the architecture and implement the video analysis and post-processing system we designed. Figure 2 illustrates the system's two components and the associated main steps we designed to build and implement the proposed system.

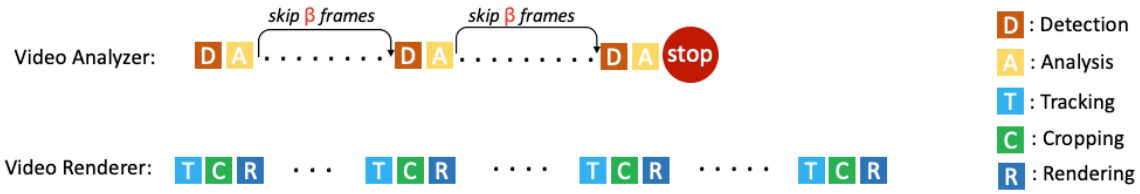


Figure 2: The video analysis and post-processing system

The video analyzer consists of the object detector and the analyzer. First, we trained a model based on the architecture of Mask R-CNN [13] with Microsoft Common Objects in Context (MS COCO) [23] and pre-trained weight to fine-tune the model (transfer learning) used in the system. There are three object classes that need to be trained: face, person, and screen. We aim to extract the main instructor's torso from the video, frame by frame, for instructor detection. The MS COCO pre-trained weight is robust for detecting a person, even though only a person's back or head is shown. There are several steps to detect the main instructor in the given videos. The system combines the face detection results and the frontal torso detection results to filter the noisy objects whose backs are presented in the video from the beginning to the end. These persons are distracting compared to the main instructor. Then, we retrain a model to detect the faces and persons to find the frontal human torso. Then we proposed a novel approach to confirm the main instructor in multiple-person scenarios. For screen detection, the screen is a new object category besides the MS COCO pre-trained weight. We annotate images that contain screen objects in new datasets and feed the annotated images into the deep neural network we used in the system. In the end, the video analyzer would output the main instructor's bounding box information and detected screen.

The video renderer combines the video analyzer's output from the original video to generate a new layout video. We use image processing and video processing technology to render a new layout video with a small size and high quality. In image processing, because the screen sometimes looks skewed due to the angle of video shooting, we designed algorithms to filter contours to get the exact screen contour and find the four corners of the screen contour. Then we utilize the perspective transform algorithm to correct the skewed

screen. We also designed a video layout for objects reframing adaptive to updated screen sizes. In video processing, we utilize the Kernelized Correlation Filter (KCF) [14] [15] to track the main instructor frame by frame. It is beneficial to reduce the jitter of a person in the newly-generated video. Finally, the system combines the reorganized frames and audio, separated before detection, to render a new layout video.

3.2 Video Analyzer

In this part, we introduce the architecture of the video analyzer (Figure 3). There are several components in the video analyzer: object detector (3.2.1, 3.2.2), instructor and screen filter (3.2.3), main instructor classifier, and the Region of Interests (ROI) selector (3.2.4).

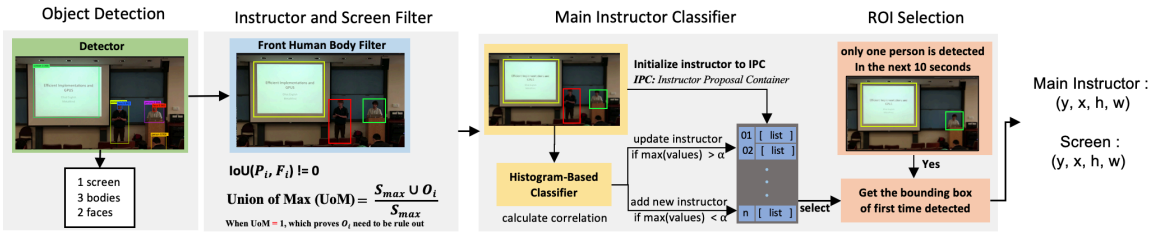


Figure 3: The architecture of the video analyzer

3.2.1 Object Detection Framework

The Convolutional Neural Network (CNN) has been commonly utilized to extract features in object detection tasks using various object detection models in the real world.

There are two main types of object detection frameworks based on CNN: a one-stage framework and a two-stage framework. The one-stage framework, including YOLO

V3 [16] and SSD [17], are fast in training with low accuracy. The two-stage frameworks, such as Faster R-CNN [18] and Mask R-CNN [13], are slow in training with high accuracy.

Considering the real shooting scenarios, we focus on medium objects (whose sizes are in the range of 32^2 and 96^2 at pixel level) and large objects (whose sizes are in the range greater than 96^2 at pixel level). Moreover, we utilize the framework with high mean Average Precision (mAP) of the bounding boxes to detect the main instructor and screen.

We compared four popular frameworks (Mask R-CNN [13], Faster R-CNN [18], SSD513 [17] and YOLO V3 [16]) on the MS COCO dataset. The standard COCO metrics included mAP, Average Precision for small objects (AP_S), Average Precision for medium objects (AP_M), and Average Precision for larger objects (AP_L) [23]. We use Mask R-CNN as the object detection model because of its high mAP in detecting both large objects and medium objects; besides, the mask-branch in Mask R-CNN can help increase the detection accuracy. In this research, we train the object detection model based on the Mask R-CNN framework.

3.2.2 Mask R-CNN Architecture

Mask R-CNN (regional convolutional neural network) [13] is a two-stage object detection framework. In the first stage, the model scans the image and generates proposals (areas that likely contain an object). In the second stage, the model classifies the proposals and generates bounding boxes and masks. Both stages are connected to the backbone structure.

ResNet101 is an FPN-style deep neural network consisting of a bottom-up pathway, a top-bottom pathway, and lateral connections. FPN outperforms other single convolution neural networks mainly because it maintains strong semantical features at various resolution scales. The Region Proposal Network (RPN) is a lightweight neural network. It scans all FPN top-bottom pathways and proposes regions containing objects.

In this work, ResNet101, paired with a Feature Pyramid Network (FPN) [19], is utilized as the backbone to extract features from input images. The early layers detect low-level features (edges and corners), and later layers successively detect higher-level features (face, person, screen). The input image is converted from 1024 x 1024 x 3 (RGB) to a feature map in the shape of 32 x 32 x 2048 after the backbone network processes the image.

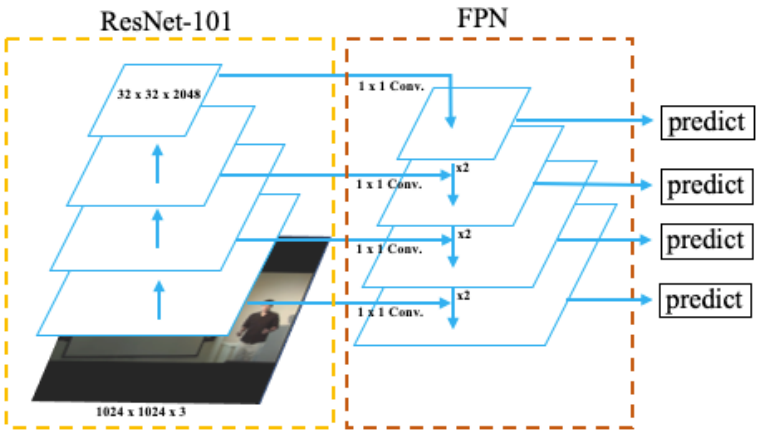


Figure 4: The backbones of ResNet-101 and FPN

In the second stage, ROIAlign is used to locate the feature map's relevant areas, and a branch is used to generate one mask for each object at the pixel level. We keep mask-branch in the architecture because it is beneficial to increase the object detection accuracy. The Softmax classifier is used for multi-target classification. The Softmax layer in the detection model is retained to get the corresponding probabilities of the three different categories.

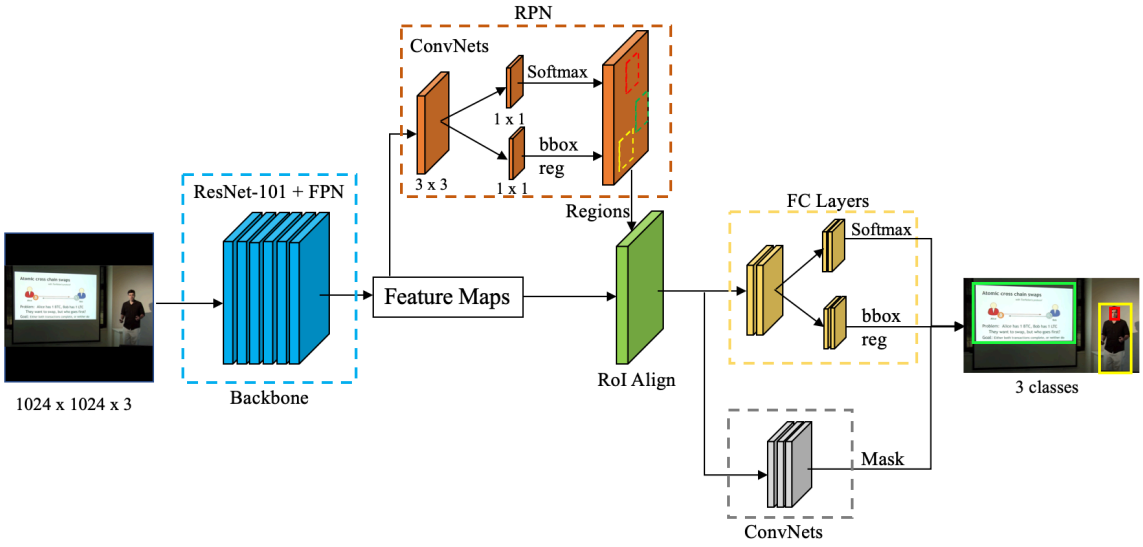


Figure 5: The Mask R-CNN Architecture

3.2.3 Instructor and Screen Filter

A lecture video recorded in the classroom contains many objects, usually more than one person or one screen. Figure 6a shows the detection results. In this case, the system is designed to filter out irrelevant objects. In Figure 6b, the system filters out some people

who present their backs in the video. In Figure 6c, the system filters out some objects that were detected on the screen.

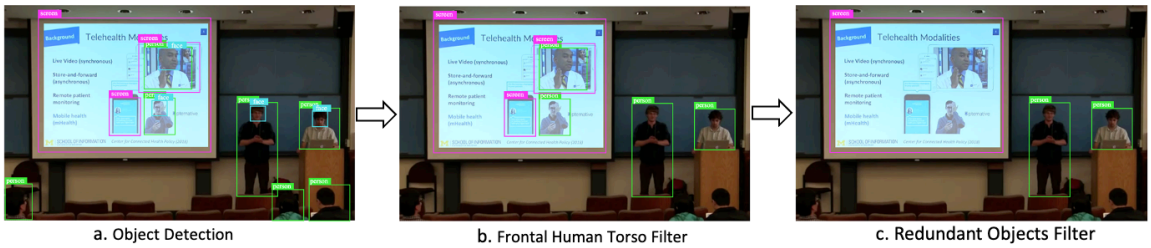


Figure 6: The Process of Filtering the Instructor and Screen

Frontal Human Torso Filter: We trained an object detector to recognize and present bounding boxes for faces and persons in the video. According to the detection results, we verified the relationship between each detected person and face. The IoU formula is proposed to calculate if each person's bounding box P_i overlaps with a face's bounding box F_j. If the value of IoU does not equal zero, it means that the person presents his front torso in the video. In this way, the system saves all persons with frontal torso and filters out all persons with their back to the screen.

Intersection over Union (IoU)

$$\text{IoU} = \frac{P_i \cap F_j}{P_i \cup F_j}$$

where the object P_i should be keep, If $\text{IoU} \neq 0$

Redundant Objects Filter Out of the Screen: We proposed an approach to filter out redundant and irrelevant objects using a parameter named the Union of Max (UoM). The (UoM) is calculated as the union of the area (S_{max}) of the screen and the area (O_i) of the object over the area of S_{max}. If the UoM is equal to one, then the object (O_i) is fully overlapped with the screen (S_{max}).

Union of Max (UoM)

$$\mathbf{UoM} = \frac{S_{max} \cup O_i}{S_{max}}$$

where the redundant object ($\boldsymbol{O_i}$) is filtered out (if UoM=1)

Another problem was more than one instructor in the video when the system was filtering out some irrelevant objects. It is hard for the system to tell which person is the main instructor. To solve this problem, we proposed the following main Instructor classifier.

3.2.4 Main Instructor Classifier

As noted in Section 2.1, the main instructor is the person the audience is supposed to pay attention to. It is challenging for the system to recognize the main instructor in the following scenarios:

1) More than one person is presented in a frontal view. In this case, the system has no idea how to detect which person is the main instructor.

2) Some people presenting the frontal view may pop up shortly in the video.

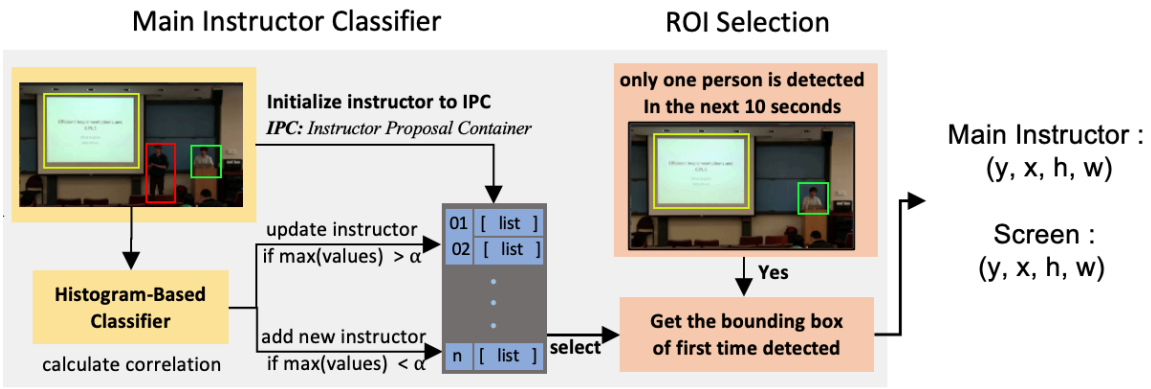


Figure 7: Overview of identifying the main instructor. Given a video, the proposed approach generates the Instructor Proposed Container (IPC) that map unique instructor identity to an instructor feature list that includes feature, times, and ROI. The system will stop until the detector detects only one instructor in the following consecutive 10 seconds.

The goal here is to recognize and identify the main instructor using visuals in Figure 7. The object detection model is utilized to detect the face, person and screen in the experiments. After the object detection, the new results are run frame by frame. For each result, we used the instructor filter (Sec 3.2.2) to determine the front human torso. Once a front human torso is determined in this process, the detected result will be stored in the Instructor Proposed Container (IPC). The IPC structure is similar to a hash table that can map a unique identity (ID) to an instructor information list that includes features, times, and ROIs. The IDs represent the keys in the hash table. The instructor information is stored

as values. As a result, the system can search the instructor features in the following video frame.

We then use a histogram-based method to calculate the correlation between the previous instructor's features and the current instructor's features.

Histogram-Based Classifier: This is a method to compare the similarity between two different size images. The three reasons we utilize a histogram-based classifier are:

1) Different instructors wear clothes in various styles and colors. We choose the instructors' clothes as distinguishing features.

2) Slight differences may exist between two consecutive frames in terms of the same instructor;

3) This method consumes few computing resources.

The equation of computing correlation between two images is shown below:

$$d(H_1, H_2) = \frac{\sum_I \left(H_1(I) - \overline{H1}\right)\left(H_2(I) - \overline{H2}\right)}{\sqrt{\sum_I \left(H_1(I) - \overline{H1}\right)^2 \sum_I \left(H_2(I) - \overline{H2}\right)^2}}$$

where

$$\overline{Hk} = \frac{1}{N}\sum_J H_k(J)$$

(N is the total number of histogram bins)

The system decides whether the instructor's detected feature should be updated in the original identity (ID) or add a new identity into IPC (we set the default threshold α as 0.5). The system calculates each front human torso's correlation value with each instructor in the IPC using the histogram-based classifier after each torso being filtered. And then find the maximum value from the obtained calculated values. If the maximum correlation value is higher than α, the current instructor's feature will be updated into the existing corresponding instructor's information list. Otherwise, a new instructor identity (ID) and corresponding information list will be added to IPC. At the same time, the system will determine that if only one instructor is detected in consecutive 10 seconds in the video, the detector will stop detection. In the end, the analyzer will output the bounding box of the main instructor.

In summary, we successfully solve the two problems presented at the beginning of this part. Besides, our system uses less memory when updating the instructor’s feature of each frame. It uses fewer computing resources by using the histogram-based verification algorithm.

3.3 Video Renderer

The video renderer will rerun the original video from the beginning with the bounding boxes of the main instructor and screen generated from the analyzer (Figure 8). Due to different shooting conditions, the screen’s angles in the video are different. It is necessary to correct the skewed screen presented in each frame. We describe the approach

for screen contour detection (Sec. 3.3.1), screen contour filter (Sec. 3.3.2), screen viewpoint correction (Sec. 3.3.3), and adaptive object reorganization (Sec. 3.3.4).

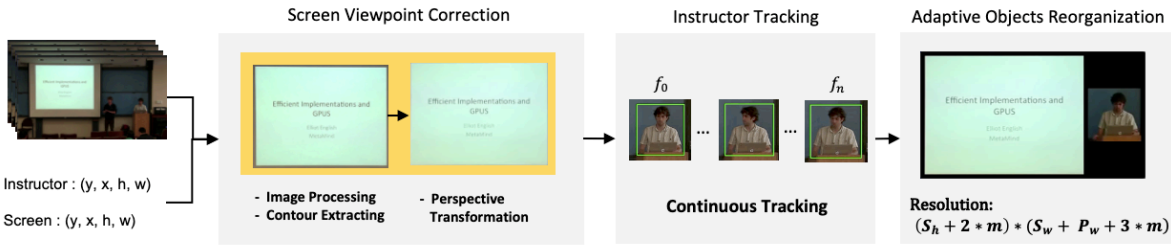


Figure 8: The Video Renderer Architecture

3.3.1 Screen Contour Detection

After the model detection, the system gets the bounding box information [y, x, h, w] for each object from detection return values, which means the top-left coordinate point (x, y), height, and width of the bounding box.

The four points of a rectangular screen may construct a skewed screen. The system exacts the screen contour to correct the screen viewpoint to a rectangular one. Fortunately, the outside of the screen boundary usually has the same feature. The brightness is the main feature is selected to divide the edges.

Before detecting the screen's contour, the screen region is preprocessed because it is susceptible to noise in the image. The first step is to remove the image's noise with a 5 x 5 Gaussian filter. This also contributes to reducing the impact of some small objects for screen contour detection. The brightness channel is the image itself for grayscale images,

and no color channels (RGB) are used. The Canny edge detector [20] is then applied, which models edges as sharp discontinuities in the brightness channel, adding non-maximum suppression and hysteresis thresholding steps.

Contour is joining all the consecutive points (along the boundary) that have the same color or intensity. Then, the system uses a binary classifier to determine whether contours pass through an image pixel or not. Those operations are completed based on the OpenCV framework.

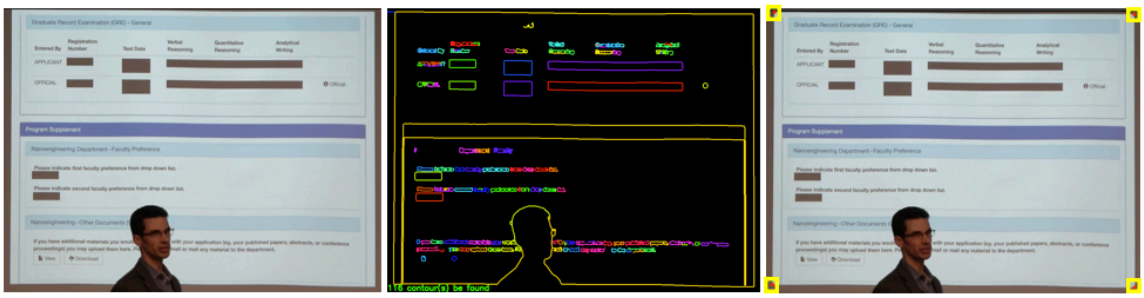
Although the Canny edge detector [20] works well, some noisy contour shapes need to be removed from the contour data. We designed an algorithm to optimize the algorithm of selecting a screen contour.

3.3.2 Screen Contours Extracting Algorithm

We designed an algorithm to filter the screen contour in terms of two features.

1) There are four corner points in a screen contour.

2) The contour of the screen always has the largest area in the detected area.



A B C

Figure 9: Image A is input. Image B is the result of contour using contour detection, and each color stands for a distinct contour; 116 contours can be found. Image C results from four corner points (the red points in the yellow box) running on our algorithm.

An example is shown in Figure 9, and the corresponding pseudo-code is shown in Algorithm 1.

Algorithm: Screen Contour Extracting

function Find-Screen-Contour (*N*):
Input: a list of Numpy array of (x, y) coordinates of boundary points of the object
Output: a list of four corner points of screen contour
if list is empty **then**
  return 0
$S_{max} = 0$
points = [(0,0), (0,0), (0,0), (0,0)]
for each Numpy array in *N* **do**
  S_A = min ($x_i + y_i$)
  S_B = min (y_i - x_i)
  S_C = max (y_i - x_i)
  S_D = max ($x_i + y_i$)
  $S_{\Delta ABC}$ = | determinant (S_B - S_A, S_C - S_A) | / 2
  $S_{\Delta DCB}$ = | determinant (S_B - S_D, S_C - S_D) | / 2
  $S = S_{\Delta ABC} + S_{\Delta DCB}$
  if $S > S_{max}$ **then**
    $S_{max} = S$
    points = [S_A, S_B, S_C, S_D]
return points

Algorithm 1. Pseudo code of screen contour extracting algorithm

Get Four Corner Points: Each contour is represented using a NumPy array of (x, y) coordinates of the object's boundary points. Assume there are (*N*) contours, and each contour is composed of a series of coordinate points. We utilize the coordinate point

operation method to get the global four corner points for each contour (S_A stands for top-left, S_B stands for top-right, S_C stands for the bottom left, S_D stands for bottom-right).

$$S_A = \min(x_i + y_i)$$

$$S_B = \min(y_i - x_i)$$

$$S_C = \max(y_i - x_i)$$

$$S_D = \max(x_i + y_i)$$

Calculate the Area: Each irregular quadrilateral area (S) is shown in Figure 9b. To get the accurate area of each contour, we divided the quadrilateral area into two triangles. We use the Vector Product Method to get the triangle area (S_Δ):

$$S_\Delta = \frac{1}{2} \left| \overrightarrow{AB} \right| * \left| \overrightarrow{AC} \right| = \frac{1}{2} |x_1 y_2 - x_2 y_1|$$

$$S = S_{\Delta ABC} + S_{\Delta DCB}$$

Based on the above two features, the system can accurately locate the screen contour. Moreover, we can utilize the four corner points to correct the screen viewpoint if the screen viewpoint is skewed.

3.3.3 Screen Viewpoint Correction

As we mentioned in Chapter 3, a skewed viewpoint of the screen will give the user a bad experience. As a result, we adopt the four corner points to correct the skewed image viewpoint. The Perspective Transformation algorithm can help us to achieve this purpose.

In a scenario where a person stands in front of the bottom-left corner or bottom-right corner, the system can quickly and correctly get the fourth point. Figures A and B show the situations in Figure 10.

Besides, the screen bounding box is a rectangle, while the screen contour is an irregular quadrangle. Although the screen and the person bounding box intersect, the bounding box containing the person does not have an intersection with the screen contour. Figures 10, C and D show these scenarios.

To solve this problem, we proposed two steps to find the points: 1) Locate the person's location; 2) Estimate the occlusion point.

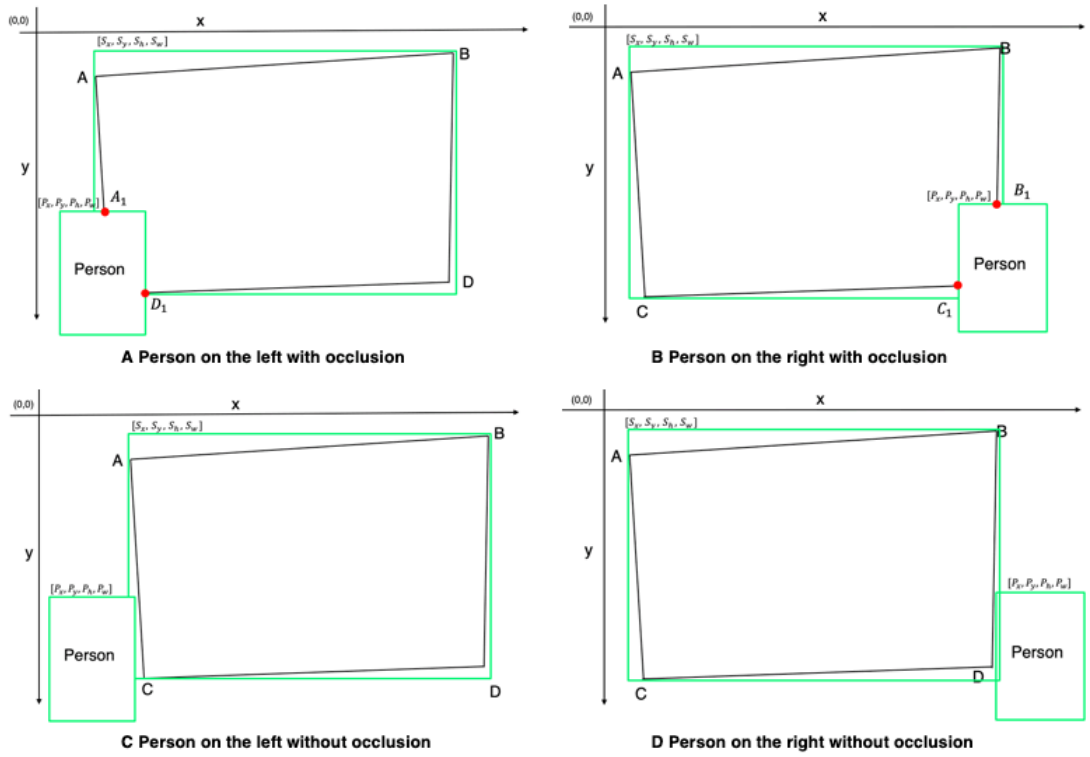


Figure 10: Figure A and B show the situations of the fourth point with occlusion; Figure C and D show the situations of the fourth point without occlusion.

Locate the person's location: First, the system gets a list of contour points and a coordinate [y, x, h, w] of the person bounding box. Then the system searches value x and value y in the list of screen contours separately.

If values x and y are not in the list of screen contours, the person's bounding box and screen contour do not intersect with each other. In this case, we exclude Figure C and Figure D. If value y is in the list of screen contours but value x is not, then the bounding box of the person and screen contour has an intersection area, and the person stands at the left corner. Besides, if both value y and value x are in the screen contour list, the bounding box of the person and screen contour has an intersection area, and the person stands at the right corner.

$$\begin{cases} P_x \; and \; P_y \; are \; not \; in \; N; & No \; intersection \; area \\ P_y \; in \; N \; and \; P_x \; not \; in \; N; & person \; at \; left \; corner \\ P_y \; and \; P_x \; are \; in \; N; & person \; at \; right \; corner \end{cases}$$

Note: This is the search logic in N to confirm where the location is the person.

Assume (N) stands for the list of screen contours, so they have the relationships below:

Based on the search logic, we can get the corresponding points that the person intersects with screen contours.

Find occlusion points: For example, the person stands on the left corner, and left corner point *C* is occluded in Figure 10A.

Assuming there are lines A and D, and each line consists of a lot of coordinates. In terms of the Linear Regression Equation, we can find the slop $\mathrm{A_s}$ and bias $\mathrm{A_b}$ for a line which goes from point $\mathrm{A_0}$ to point $\mathrm{A_1}$. Using the same principle, we can also get slop $\mathrm{D_s}$ and bias $\mathrm{D_b}$ for a line which goes from point $\mathrm{D_0}$ to point $\mathrm{D_1}$. Furthermore, the two lines always intersect at one point, which is defined as the occlusion point.

$$\begin{cases} y = A_s * x + A_b \\ y = D_s * x + D_b \end{cases}$$

So, in terms of two linear equations, we can get the occlusion point C.

3.3.4 Adaptive Objects Reorganization

Now, we get the corrected screen with a good viewpoint. The next step is to combine the screen and instructor into a new video stream.

Videos filmed and edited for television and desktop are typically created and viewed in landscape aspect ratios (16:9 or 4:3).

We designed three types of video layout:

(1) Only a screen is presented in the video.

(2) The screen is shown on the left, and the main instructor is shown on the right.

(3) The screen is on the right, and the main instructor is on the left.

The total video size is:

$$Video\ Resolution = (S_h + 2 * m) * (S_w + P_w + 3 * m)$$

$\boldsymbol{S_h}$: $screen\ height$

$\boldsymbol{S_w}$: $screen\ width$

$\boldsymbol{P_w}$: $instructor\ width$

$\boldsymbol{m}$: $screen\ margin$

CHAPTER 4

EXPERIMENTS

In this chapter, we present two parts in detail. First, we introduce the dataset in detail for the object detection model, model training, and model evaluation (Sec. 4.1). Second, we introduce datasets preparation for testing and evaluating the system performance (Sec. 4.2).

4.1 Object Detection Model

When we train a model, a dataset is essential according to the requirements of the task. Once the dataset is available, we fine-tune the model hyperparameters to obtain the best detection result with high accuracy. For the model evaluation, recent research papers tend to present models with AP (Average Precision) results in the MS COCO format. In this work, we utilize the COCO format AP to evaluate the performance of the object detection model. The model is tested on the same test set on different Intersection over Union (IoU) threshold settings. AP@[.5:.95] corresponds to the average AP for IoU from 0.5 to 0.95 with a step size of 0.05.

4.1.1 Dataset Preparation

Our dataset has 1,935 training images and 214 test images. There are three categories: face, person, and screen. There are 8,156 objects, including 6,980 images of faces, 712 images contain persons, and 464 images contain screen objects [Table 1].

We used part of the images downloaded from the WIDER FACE [21] dataset for face detection. Most images include the whole body of a person in the original dataset, and a *person's body* is defined as a negative sample during the training. First, we preprocess the dataset to keep only face features as much as possible and keep the hostile sample areas not related to a person's body (Figure 11). We ran two sets of experiments. It turns out that this method can accelerate the convergence of value loss and improve a person's body's detection precision.

Figure 11: Row 1 shows the raw image in the WIDER FACE dataset, and Row 2 shows the preprocessing image we used during training.

For the persons' dataset, the weights in the model are trained using the MS COCO pre-trained model. Hence, a small amount of person dataset is required for this task. We downloaded and annotated images containing persons from the internet.

	Total	Face	Person	Screen
Total	8,156	6,980	712	464
Train dataset	7,499	6,519	597	383
Validation dataset	357	251	65	41
Test dataset	300	210	50	40
Dataset Sources	-	WIDER FACE [21]	Download (YouTube) [22]	Download (YouTube) [22]

Table 1: The number of objects for each category in the train dataset and validation dataset

A displayed screen could be found in a TV, monitor, laptop, etc. Although many existing datasets can detect these physical media objects' outlines, as far as we know, there is no specific dataset that could be used to identify the displayed screen.

We synthesize more images to construct a larger size dataset. Because videos are shot in different places, the screen position, screen angle, and lighting conditions can vary from video to video. We downloaded some videos in different scenarios from the Internet. We also shot some videos in different places with various shooting angles, classrooms, meeting rooms, studios, etc. We utilized the OpenCV framework to generate datasets by running different videos. There are 30 frames per second in a video stream, and we set the inter-arrival time to 10 minutes to make sure one frame is different from another frame. In

the end, we got 464 images and divided them into two sub-datasets: the training set with 383 images, the validation set with 41 images and the test set with 40 images.

4.1.2 Detection Model Training

Since the datasets are small, we initialized the same training network parameters as the parameters in the MS COCO pre-trained model, which indicates the weights have already been trained and saved in the model after training tens of thousands of images for several days at the time of initialization. The training procedure is a fine-tuning process to make the predicted results more accurate in detecting persons, faces, and screens.

There are two steps in the training process. First, fine-tuning the last three branches in the Mask-RCNN model to enhance person, face, and screen feature extraction ability. Then, we trained the network on 2 GPUs (Quadro RTX 5000 16G) for 13k iteration with a mini-batch size of 4. The learning rate was initialized to 0.001, which was decreased by 50% after 2.4k iterations, 4.58k iterations, 7.2k iterations, and 9.6k iterations step by step. The weight decay rate was set to 0.0001. An SGD optimizer was utilized with a momentum of 0.9, the same as the Mask-RCNN paper's value. Figure 12 shows the convergence curve of the training. The validation dataset was utilized to verify the trained model’s performance and pick the best model.

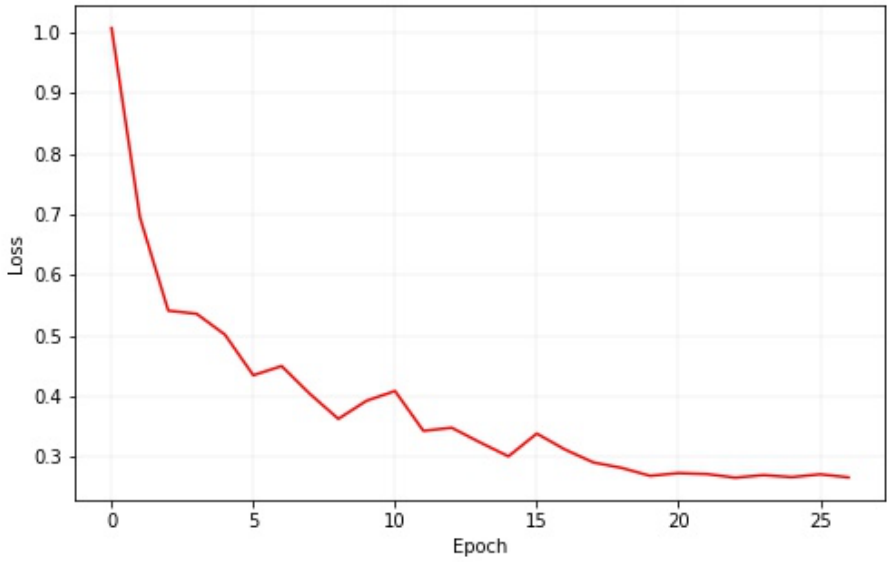


Figure 12: Epoch vs. Loss

4.1.3 Detection Model Evaluation

We evaluated the object detection model on frontal human torso detection, instructor selection, and screen detection. Higher IoU means the output of the model is closer to ground-truth values. A higher value of the AP means the model detection result is improved.

Frontal Human Torso Detection: All the test dataset images were fed into the detection model to evaluate the performance of the frontal human object detection model. The mAP values are shown in Figure 13. The AP values are set to 10 IoU thresholds (from 0.50 to

0.95, increasing at a step of 0.05). The positive samples and negative samples in detecting the frontal human are shown in Figure 14.

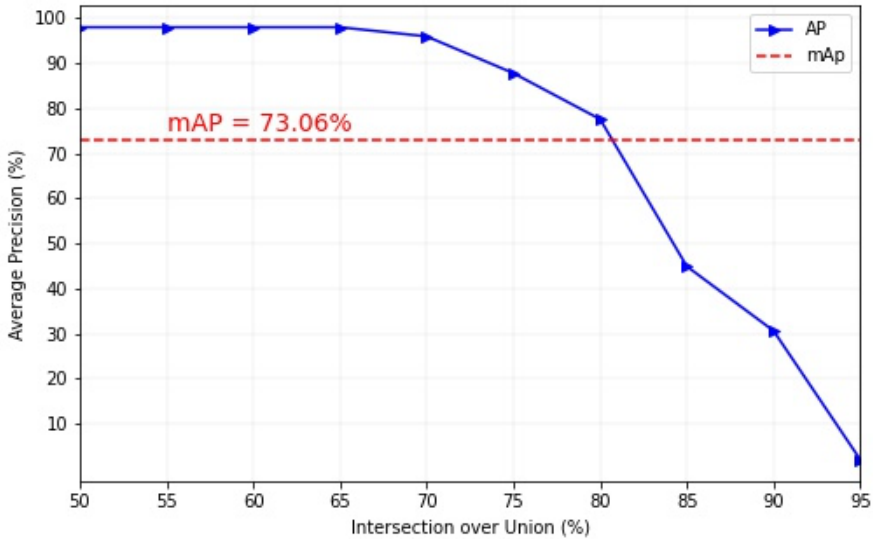


Figure 13: Average Precision vs. Intersection over Union

Figure 14: The test result for frontal human. The green box shows the positive result. The red box shows the negative result.

Screen Detection: All the test dataset images were fed into the detection model to evaluate the performance of the screen object detection model. The mAP values are shown in Figure 15. The positive samples and negative samples in detecting the screen are shown in Figure 16. The AP values are set to 10 IoU thresholds (from 0.50 to 0.95, increasing at a step of 0.05).

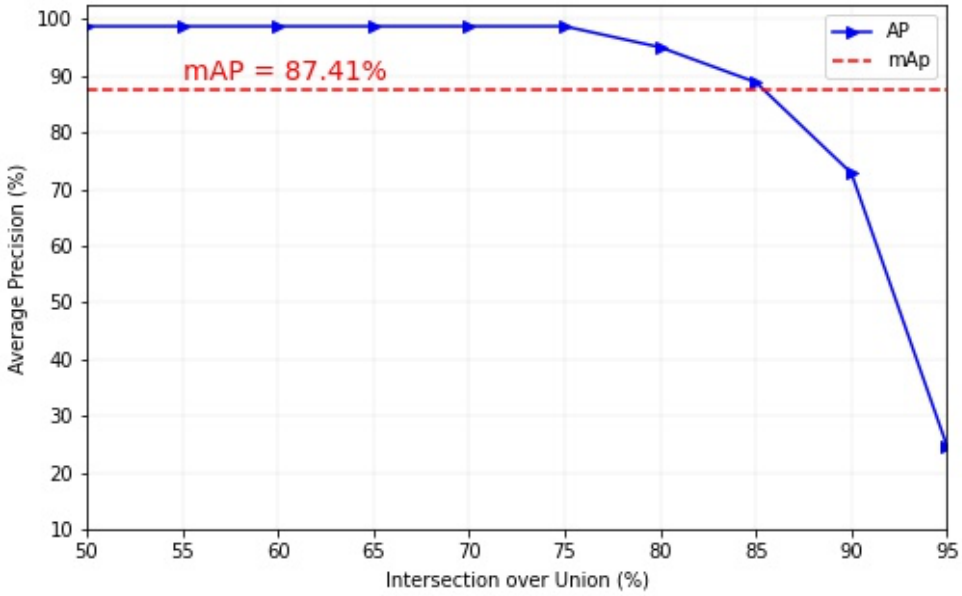


Figure 15: Average Precision vs. Intersection over Union

Figure 16: Screen detection results in different situations

Overall, the frontal human detection gets a mAP of 73.06%, and the screen detection gets a mAP of 87.41%.

We observed three reasons for high mAP values: 1) The classification task is too simple to classify person, face, and screen. In multi-task object detection, the AP may get lower. 2) The screen is such a large target that it almost occupies the image. Since detecting large objects is more accessible, the AP value becomes higher after averaging all the results. In this case, a larger dataset is required. 3) The image scenarios are similar, mainly concentrating on the classrooms or meeting rooms. The AP value goes up since many scenes between the training dataset and test dataset overlap. More images from various scenes are required for both the training set and the test set.

4.2 Metrics for Evaluating the System Performance

There are a total of 30 videos for testing the performance of the system we proposed. We used six videos of different scenes and then converted each video to 5 different resolutions. We analyzed the video analysis time, video rendering time, average analysis time between different frames per detection, and average video file size in different video resolutions.

4.2.1 Dataset Preparation

We collected six scenes of videos from the Internet [22] and then preprocessed the videos. First, we shortened each video's playback time to 10 minutes and then generated five different resolutions (320*240, 480*352, 640*480, 960*720, 1120*832) separately.

Figure 17: Video Source [22]

4.2.1 System Performance Evaluation

We evaluated the system's performance using the average analysis time, the average rendering time, the average total generated time, frame per detection, and the average output file size. The videos were tested on a CPU and GPU individually.

In Figure 18, we observed that more extensive resolution videos usually require more time to be analyzed. As the video resolution increases, the average deviation of the analysis time becomes larger, and higher resolution videos require higher model detection accuracy. GPU analysis time is less than CPU analysis time.

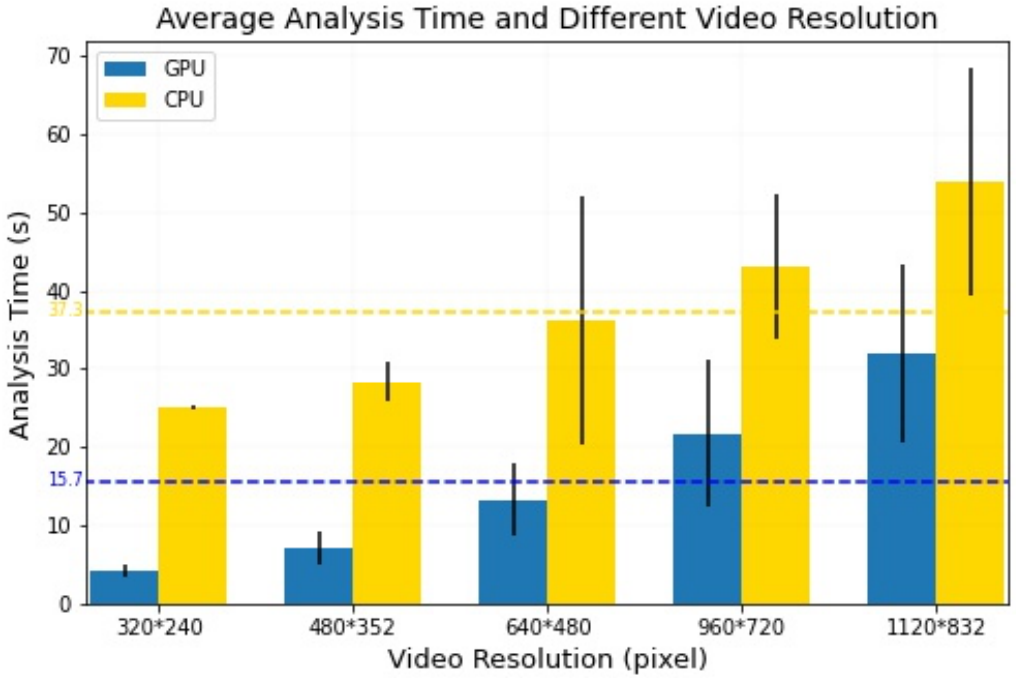


Figure 18: Average Analysis Time

In Figure 19, we observed that larger resolution videos usually require more time to be rendered. The GPU rendering time is almost the same as the CPU rendering time, indicating that rendering a video does not require higher performance computing resources.

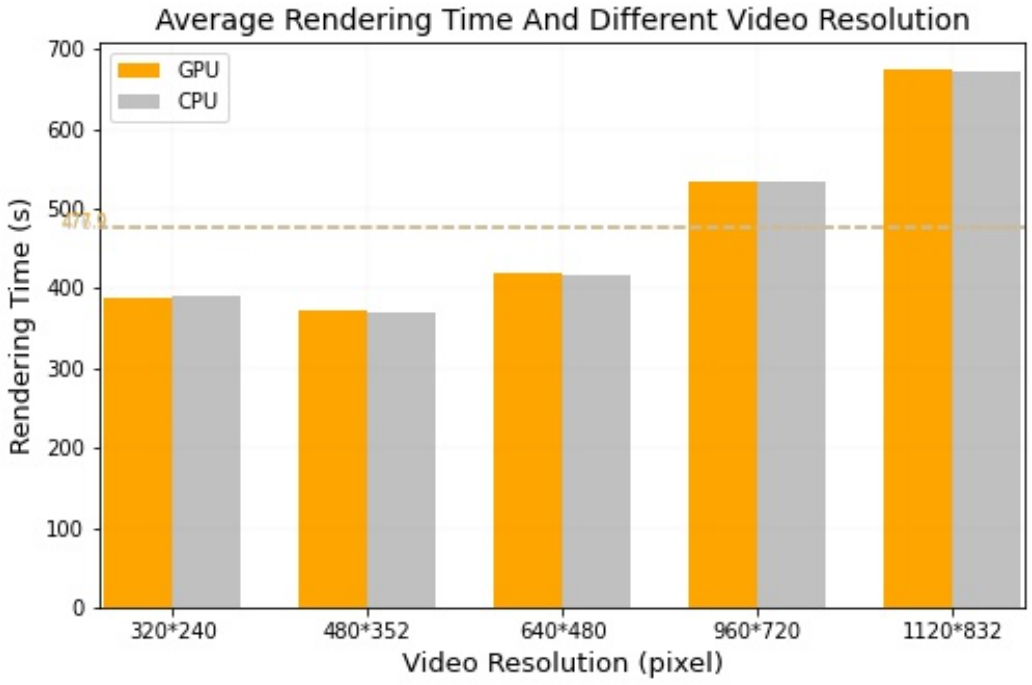


Figure 19: Average Render Time

From Figure 20, We found that as the video resolution increases, the total time to generate a new video will be greater than the video playtime, proving that if we need to generate new videos faster, we need more computing resources.

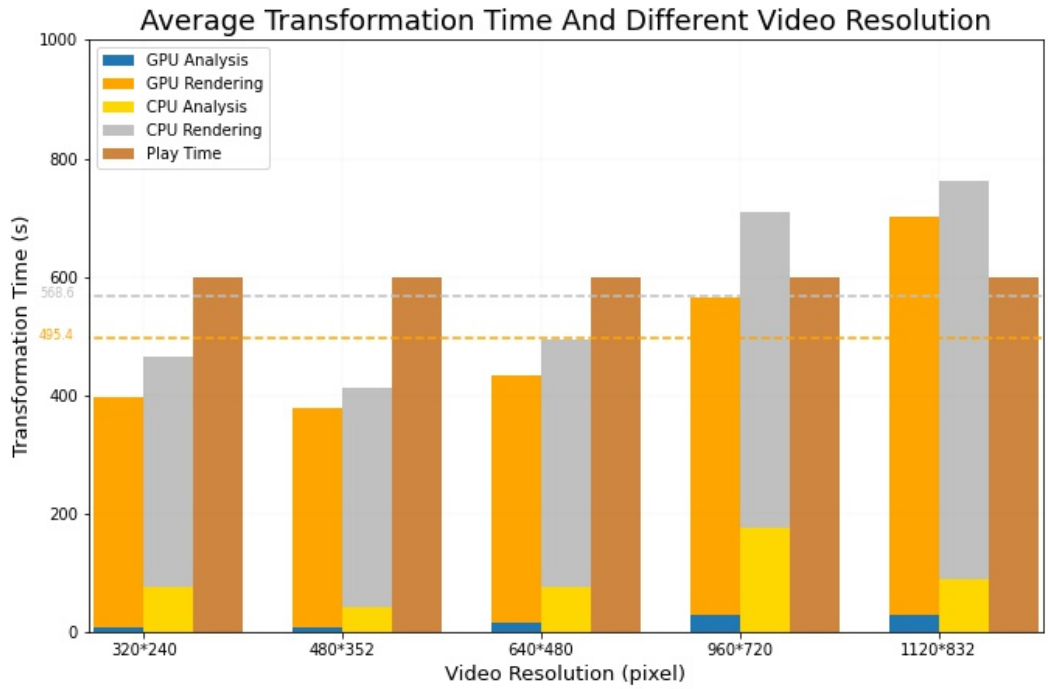


Figure 20: Average transformation time

In Figure 21, we observed that more analysis time is required to complete the task when the detection interval time is increased. This can be explained using a mathematical formula.

$$\begin{cases} \boldsymbol{d}: time\ of\ frame\ detection \\ \boldsymbol{t}: time\ of\ only\ one\ person\ is\ detected \\ \boldsymbol{s}: time\ of\ skip\ undetected\ frames \end{cases}$$

$$Analysis\ Time = d * t * s$$

In this formula, the detection time ***d*** is known as a constant value. If we adjust the frame per detection to a higher value, ***s*** will take a longer time to skip undetected frames, and ***t*** also takes more time. Hence, we suggested adjusting the frame per detection to 20.

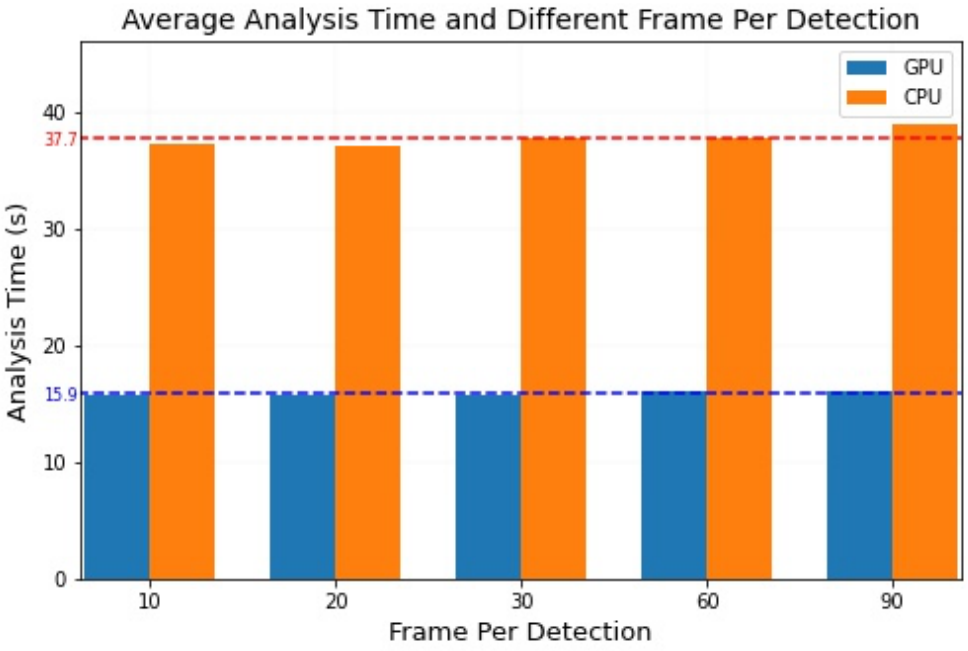


Figure 21: Average analysis time

In Figure 22, we observed that the average video file size was reduced by 50%. This finding is beneficial for users.

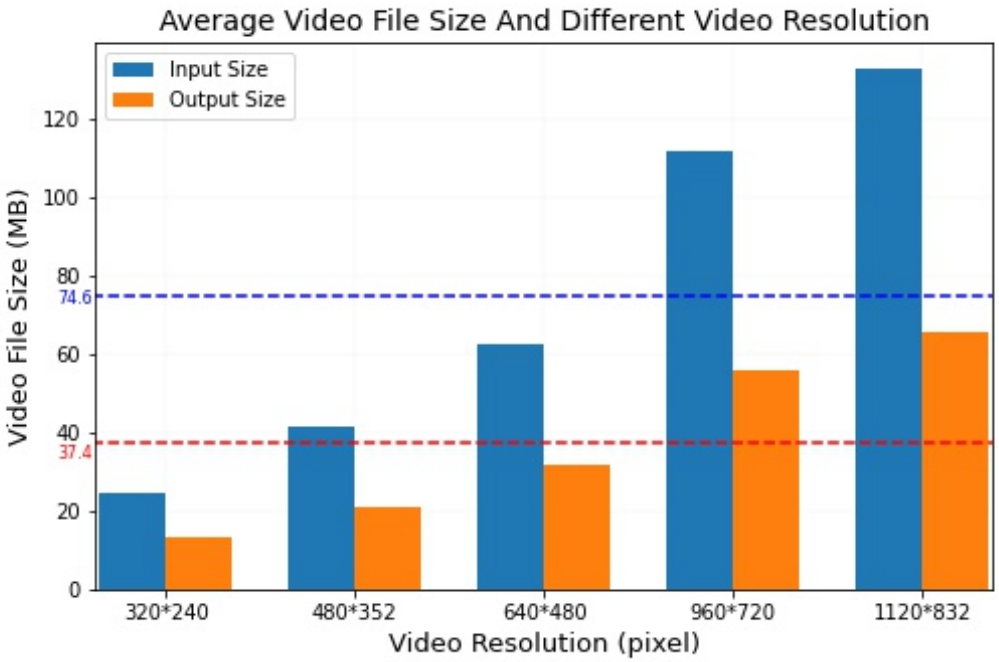


Figure 22: Average video size

Our system takes a short analysis time to determine the main instructor and screen. And we designed an automatic post-processing system to improve the users' watching experience and reduce post-processing time, costs, and video file size. Although our system's rendering time is longer than the video's playtime, it is much faster than manual post-processing.

CHAPTER 5

CONCLUSION

We have presented an intelligent analysis and post-processing system to analyze and post-process the lecture videos. Using this system is time-efficient to convert a raw video to a new layout video to reduce post-production costs and give the audience an enjoyable watching experience. At the same time, the system also helps to reduce the video file size. We built the software pipeline system, trained the modified Mask-RCNN model on a new dataset, proposed a new approach to recognize and detect the main instructor, and designed a new algorithm to find screen contour and correct the video's skewed screen. Experimental results show that the system works well in many scenarios. The improvement and application of this system can be conducted in the future:

- The object detection model is used to recognize and detect the screen region in a video. The detection result lost some information when the IoU ratio was larger than 85%. In deep learning, a richer dataset usually contributes to a better result in addressing overfit. The dataset used in this work can be further extended in several ways: taking more photos under different illuminations and viewpoints, using position augmentation (flipping, cropping, affine transformation, etc.), and color techniques (brightness, contrast, etc.) to generate more images, etc.
- For the main instructor identification, we use the frontal human torso feature to identify the main instructor with time cues. Under multiple-instructor conditions, this method is limited to focus on a fixed instructor. Lip motion detection could be

added to identify the main instructor at a given time point in future research. This change will help make the generated video information more comprehensive and more vivid.

- We implemented our system on a server with two GPUs. If we run our system on a PC-level CPU, the whole process would cost less time. Besides, the system is not adaptive in a real-time scenario. The main instructor detection and the detection speed are bottlenecks in a live video stream. New techniques could be explored to achieve real-time rendering.
- The evolution of object detection frameworks is always beyond our imagination. A more efficient and adaptive framework could help improve object detection accuracy and speed in the future.
- We also proposed a framework for real-time video transformation (Figure 23) for researchers interested in this domain for future reference.

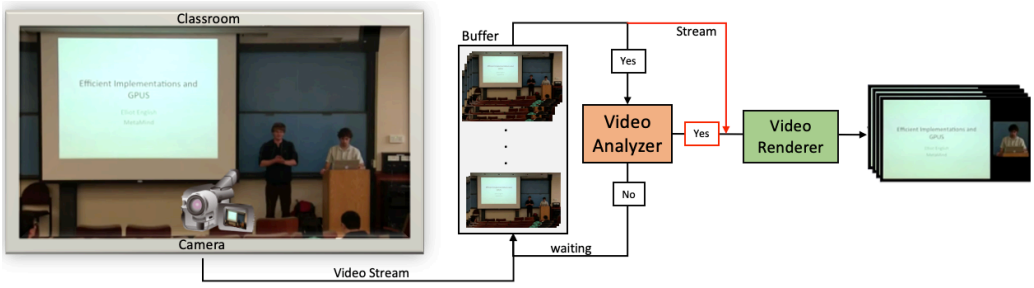


Figure 23: The architecture of real-time analysis and post-processing system

BIBLIOGRAPHY

[1] Bo Peng and Gang Qian, “Binocular Dance Pose Recognition and Body Orientation Estimation via Multilinear Analysis,” in Conf. 2008 Computer Vision and Pattern Recognition (CVPR).

[2] Angelo Maria Sabatini, “Estimating Three-Dimensional Orientation of Human Body Parts by Inertial/Magnetic Sensing,” in Sensors 2011, 11(2), 1489-1525

[3] Cheng Chen, Alexandre Heili, and Jean-Marc Odobez, “A Joint Estimation of Head and Body Orientation Cues in Surveillance Video,” in Conf. 2011 IEEE international conference on Computer Vision(ICCV) Workshops.

[4] Kai-Chi Chan, Cheng-Kok Koh, and C. S. George Lee, “A 3D-Point-Cloud Feature for Human-Pose Estimation,” in Conf. 2013 IEEE International Conference on Robotics and Automation (ICRA).

[5] K. Yoo, "Frontal human detection based on body structure measured by using depth image," 2015 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, 2015, pp. 528-529, doi: 10.1109/ICCE.2015.7066511.

[6] Byungtae Ahn, Jaesik Park, and In So Kweon, “Real-time Head Orientation from a Monocular Camera using Deep Neural Network,” in Conf. 2014 Asian Conference on Computer Vision (ACCV)

[7] Jinyoung Choi, Beom-Jin Lee, Byoung-Tak Zhang, “Human Body Orientation Estimation using Convolutional Neural Network” in Conf. 2016 Computer Vision and Pattern Recognition (cs.CV).

[8] M. Everingham, J. Sivic, and A. Zisserman.” Hello! My name is... Buffy” Automatic naming of characters in TV video. In Proc. BMVC, 2006.

[9] J. Sivic, M. Everingham, and A. Zisserman. “Who are you?” – Learning person specific classifiers from video. In CVPR, 2009.

[10] M. Tapaswi, M. Bauml, and R. Stiefelhagen. ”Knock! Knock! Who is it?” Probabilistic person identification in TV series. In Proc. CVPR, 2012.

[11] Arsha Nagrani, Andrew Zisserman. From Benedict Cumberbatch to Sherlock Holmes: Character Identification in TV series without a Script. In CVPR, 2018

[12] Nagrani, J. S. Chung, and A. Zisserman. Voxceleb: a large-scale instructor identification dataset. In INTERSPEECH, 2017.

[13] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask R-CNN,” in 2017 IEEE International Conference on Computer Vision (ICCV), December 2017, pp. 2980-2988.

[14] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In proceedings of the European Conference on Computer Vision, 2012.

[15] J. F. Henriques, R. Caseiro, P. Martins and J. Batista, "High-Speed Tracking with Kernelized Correlation Filters," in IEEE Transactions on Pat-tern Analysis and Machine Intelligence, vol. 37, no. 3, pp. 583-596, 1 March 2015, doi: 10.1109/TPAMI.2014.2345390.

[16] J. Redmon, and A. Farhadi, "Yolo9000: Better, faster, stronger," arXiv:1612.08242, in press.

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multi-box detector," in European Conference on Computer Vision, September 2016, pp. 21-37.

[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in Advances in Neural

[19] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In ECCV. 2014.

[20] Canny J. A Computational Approach to Edge Detection. IEEE Trans. Pattern Anal. Mach. Intel. 1986; vol. 8, no. 6, p. 679-698.

[21] WIDER FACE: A Face Detection Benchmark, Multimedia Laboratory, Department of Information Engineering, The Chinese University of Hong Kong, http://shuoyang1213.me/WIDERFACE/

[22] YouTube, www.youtube.com

[23] Common Objects in Context (COCO), https://cocodataset.org/#home

[24] AutoFlip: An Open Source Framework for Intelligent Video Reframing, Google, https://ai.googleblog.com/2020/02/autoflip-open-source-framework-for.html