Fall 9-9-2021

# Long-term Video Object Detection and Tracking in Collaborative Learning Environments

Sravani Teeparthi

Sravani Teeparthi

*Candidate*

Electrical and Computer Engineering

*Department*

This thesis is approved, and it is acceptable in quality and form for publication:

*Approved by the Thesis Committee:*

Marios Pattichis , Chairperson

Balasubramaniam Santhanam

Ramiro Jordan

Sylvia Celedon Pattichis

# Long-term Video Object Detection and Tracking in Collaborative Learning Environments

by

**Sravani Teeparthi**

B.Tech., Electronics and Communication Engineering, 2015

THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science
Computer Engineering

The University of New Mexico

Albuquerque, New Mexico

December, 2021

# Dedication

*To my husband and parents for their support and encouragement.*

# Acknowledgments

I would first like to thank my advisor Prof. Marios Pattichis for all his patience in advising me. His guidance and ideas were crucial for my development as a graduate student. I would like to thank my committee members for allowing me to present my work and share their valuable feedback. I would also like to acknowledge Miguel A Hombrados-Herrera, and Krishna Ashok-Poddar for their valuable insights and feedback.

This material is based upon work supported by the National Science Foundation under Grant No. 1613637 and No. 1842220. Any opinions or findings of this thesis reflect the views of the author. They do not necessarily reflect the views of NSF.

# Long-term Video Object Detection and Tracking in Collaborative Learning Environments

by

**Sravani Teeparthi**

B.Tech., Electronics and Communication Engineering, 2015

M.S., Computer Engineering, University of New Mexico, 2021

## Abstract

Human activity recognition in videos is a challenging problem that has drawn a lot of interest, particularly when the goal requires the analysis of a large video database. The Advancing Out-of-School Learning in Mathematics and Engineering (AOLME) project provides a collaborative learning environment for middle school students to explore mathematics, computer science, and engineering by processing digital images and videos. As part of this project, around 2200 hours of video data were collected for analysis. This data was collected to understand how children learn in situations involving mathematical and programming challenges so as to recognize best teaching practices that support broadening the participation of underrepresented students in STEM fields. Because of the size of the dataset, it is hard to analyze all the videos of the dataset manually. Thus, there is a huge need for reliable computer-based methods that can detect activities of interest.

My thesis is focused on the development of accurate methods for detecting and tracking objects in collaborative learning environments in long videos ($>$ 1 hour).

Long-term object detection and tracking face fundamental challenges due to occlusion, illumination variations, and pose variations.

For collaborative learning groups, the thesis contributes robust methods for computer keyboard detection, tracking, and student hand detection. For hand detection, the thesis integrates object detection with clustering and time-projections for accurate, long-term assessment of student participation. The hand detection method was integrated into a writing detection system and can also be used for later research on recognizing student gestures.

All the models are validated on videos from 7 different sessions, ranging from 45 minutes to 90 minutes. The keyboard detector achieved a very high average precision (AP) of 92% at 0.5 intersection over union (IoU). Furthermore, a combined system of the detector with a fast tracker KCF (159fps) was developed so that the algorithm runs significantly faster without sacrificing accuracy. For a video of 23 minutes having resolution 858 $\times$ 480 @ 30 fps, the detection alone runs at 4.7$\times$the real-time, and the combined algorithm runs at 21$\times$the real-time for an average IoU of 0.84 and 0.82, respectively. The hand detector achieved average precision (AP) of 72% at 0.5 intersection over union (IoU). The detection results were improved to 81% using optimal data augmentation parameters. The hand detector runs at 4.7$\times$the real-time with AP of 81% at 0.5 intersection over union. The hand detection method was integrated with projections and clustering for accurate proposal generation. This approach reduced the number of false-positive hand detections by 80% by improving IoU ratios from 0.2 to 0.5. The overall hand detection system runs at 4$\times$the real-time, capturing all the activity regions of the current collaborative group.

# Contents

# List of Figures

# List of Tables

# Glossary

| | |
|---|---|
| IoU Ratio | Intersection over Union is an evaluation metric used to measure the accuracy of an object detector on a particular dataset. The IoU is the ratio of the overlapping area of ground truth and predicted area to the total area. |
| AOLME | The Advancing Out-of-School Learning in Mathematics and Engineering research study. |
| SOTA | State of the Art. |
| Binary Image | An image consisting only of black and white values. |
| Centroid | The center point in a countour. |
| Ground Truth | A set of labelled data that serves as a point of comparison |
| CNN | Convolutional Neural Network. |
| Precision | Fraction of relevant instances among all retrieved instances |
| Recall | Fraction of retrieved instances among all relevant instances |
| AP | Performance evaluation terms computed for each category. AP is the mean of the precision scores after each relevant document is retrieved. |

| | |
|---|---|
| AR | Average Recall. |
| TP, FP | True Positive, False Positive |
| TN, FN | True Negative, False Negative |

# Chapter 1

# Introduction

Object detection and tracking have advanced significantly over the last decade. It is possible to detect multiple objects in an image accurately thanks to advancements in deep learning [1]. Even though it works for images, accurate detection and tracking proved quite challenging for videos. In addition to image object detection problems, such as occlusion, scaling, variation in lighting, camera angle, we also face other video-related problems. These problems include but are not limited to (i) a large number of frames (100K images per hour), (ii) disappearance and reappearance of an object, and (iii) dynamic object pose variation.

We illustrate some of the problems with the dataset in Fig. 1.1. We have (i) different camera angles in Fig. 1.1c, 1.1b and 1.1n (ii) inconsistent illumination in Fig. 1.1a and 1.1d (iii) monitor occluding activity regions in Fig. 1.1l and 1.1o (iv) person blocking other people in Fig. 1.1n (v) activity not associated with primary table of focus in Fig. 1.1l and 1.1c (vi) other groups working in background in Fig. 1.1l and 1.1c (vii) people walking around in Fig. 1.1n and 1.1f. All the problems described above will have an impact on detection results. So there is strong interest in developing methods to analyze videos that can address the challenges of these collaborative learning environments.

## 1.1 Motivation

A significant focus of the AOLME project is to understand how students learn. Therefore, how students interact with each other, the facilitator and their lessons are of importance. To illustrate the problem, students learn and interact using their hands by typing, writing and pointing to things. For example, Figs. 1.1d, 1.1g, 1.1m, and 1.1o show students using keyboard; Figs. 1.1a - 1.1b show writing, Figs. 1.1d, 1.1g, 1.1k, 1.1m show typing, Figs. 1.1o shows students using mouse, Figs. 1.1d, 1.1d, 1.1h, 1.1i, and 1.1j show interaction using hands, Figs. 1.1f and 1.1i show gestures using hands.

The primary motivation of this thesis is to develop a robust method to detect and track objects in long videos and propose segmented hand regions in videos eliminating the background regions. This thesis will focus on detecting and tracking keyboards and hands in long videos captured in the collaborative learning environment. The thesis also proposes to use characteristics of video to provide a fast and robust system. In the future, the proposed method can easily integrate into an activity detection system providing spatio-temporal instances of typing and writing.

## 1.2 Thesis Statement

My thesis is that I can develop fast and effective object detection and tracking of hands and keyboards in long videos in collaborative learning environments through the integration of clustering, projections, tracking, and current object detection methods.

## 1.3 Contributions

The contributions of this thesis include:

1. Robust detection and fast-tracking of the keyboard in long videos (45 minutes to 90 minutes).
2. Data augmentation study to determine optimal parameters to improve hand detection.
3. A novel method that uses projections and cluster based segmentation to detect hand regions in a video.

## 1.4 Overview

The remainder of the thesis is organized into 5 chapters:

- **Chapter 2: Background.** This chapter describes prior work.
- **Chapter 3: Dataset.** This chapter describes Dataset Organization and Ground Truth.
- **Chapter 3: Methods.** This chapter describes methods for keyboard detection, tracking, hand detection, and proposal regions.
- **Chapter 4: Results.** This chapter provides a summary of results for keyboard detection, tracking, hand detection, and proposal regions.
- **Chapter 5: Conclusion and future work.** This chapter provides a summary of the thesis and recommendations for future work.

(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

(m) (n) (o)

Figure 1.1: Frames from different videos showing challenges involved.

# Chapter 2

# Background

This section provides a summary of prior research at ivPCL lab, object detection and tracking methods along with the common datasets used respectively.

## 2.1 Prior Work

This thesis derives inspiration and expands on the prior research at the ivPCL lab. In this section, we provide a summary and contrast against the prior work. Table 2.1 provides the summary.

The thesis extends prior research by developing fast and reliable methods for hand and keyboard detection that work on long videos. For keyboard detection, the thesis integrates fast tracking into the approach. Furthermore, the thesis integrates the use of projections every 12 seconds, cluster-based segmentation, and small area removal to detect hand regions and reject background images. The developed methods are tested on 7 long video sessions.

Table 2.1: Computer assisted video analysis research related to video object detection and activity detection in the image and video processing and communications lab (ivpcl).

| **Author** | **Title** | **Summary** |
|---|---|---|
| | **Computer-Assisted Video Analysis Methods** | |
| Tapia, L.S., et, al. 2021 [45] | Bilingual Speech Recognition by Estimating Speaker Geometry from Video Data | This paper proposed an approach for applying interactive video analysis system to estimate the 3D speaker geometry for realistic audio simulations. |
| Jatla, V., et, al. 2021 [27] | Long-term Human Video Activity Quantification of Student Participation | This paper proposes an approach to solve the problem of reliable video activity recognition over long videos to quantify student participation in collaborative learning environments (45 minutes to 2 hours). It also introduces a family of low-parameter 3D ConvNet architectures. |
| Teeparthi, S., et, al. 2021 [47] | Fast Hand Detection in Collaborative Learning Environments | This paper proposes a fast approach for hand detection in AOLME videos. It integrates object detection, followed by time projections, clustering and small region removal to provide effective hand detection over long videos. It also addresses the problem of occlusions and dramatic changes in appearances for an object. |
| Ulloa, A., et, al. 2021[6] | Deep-learning-assisted analysis of echocardiographic videos improves predictions of all-cause mortality | Analyzes echocardiographic videos to assist cardiologists in predicting one-year all-cause mortality. This framework method successfully increased the sensitivity of cardiologists by 13%. |
| Shi, W., et, al. 2021 [42] | Talking Detection in Collaborative Learning Environments | The paper presented a new method using motion vectors projection to detect talking combined with head detection in collaborative learning environment videos. |
| Shi, W., et, al. 2021 [41] | Person Detection in Collaborative Group Learning Environments Using Multiple Representations | The paper introduced problem with detecting groups of students from classroom videos with different angles and long videos which ranges from 1 to 2 hours and proposed a method using AM-FM representation to solve. |
| Tran, P., et, al. 2021 [48] | Facial Recognition in Collaborative Learning Videos | The paper presented fast approach to face recognition in collaborative learning environments. The method showed improvement on dealing with multiple poses and occlusions in addition to using face prototypes with k-means and sparse sampling to boost accuracy and reduce recognition time. |

| | | |
|---|---|---|
| Tapia, L., et, al. 2020 [46] | The Importance of the Instantaneous Phase for Face Detection using Simple Convolutional Neural Networks | Investigates the use of low-complexity image processing system to study advantages of using AM-FM representations versus raw images. This framework showed significant advantages by reducing training time per epoch with low-complexity architecture at a comparable accuracy using FM images for training. |
| Esakki, G., et, al. 2020 [13] [17] | Adaptive Encoding for Constrained Video Delivery in HEVC, VP9, AV1 and VVC Compression Standards and Adaptation to Video Content | Provides optimal `QP` for compressing videos for high and acceptable video streaming. For further information please refer to [16, 14, 15]. |
| Kent, R. B., st, al. 2020 [30] | Design, Implementation, and Analysis of High-Speed Single-Stage N-Sorters and N-Filters | Provides an implementation in FPGA that can greatly speedup $2 \times 2$ and $3 \times 3$ max pooling. |
| Carranza, C., et, al. 2020 [5] | Fast and Scalable 2D Convolutions and Cross-correlations for Processing Image Databases and Videos on CPUs | Proposed a method which maximizes throughput through the use of vector-based memory I/O and optimized 2D FFT libraries that run on all available physical cores. Also shows decomposition of arbitrarily large image into small using overlap-and-add. This approach outperforms Tensorflow for $5\times5$ kernels and significantly outperforms Tensorflow for $11 \times 11$ kernels. |
| Darsey, C.J., 2018.[8] | Hand Movement Detection in Collaborative Learning Environment Videos | Explores hand movement detection using color and optical flow. This approach used patch color classification, space-time patches of video, and histogram of optical flow. This approach achieved accuracy of 84% and ROC AUC of 89% on video patches from 15 video clips. |
| Jacoby, A. R., et, al. 2017 [23], [24] | Context-sensitive human activity classification in collaborative learning environments | Explores activity detection of writing, typing and talking. The method was tested on simulated data having 620 video frames for writing, 1050 for typing, and 1755 frames for talking. |
| Eilar, C. W., et, al. 2016 [12], [11] | Distributed video analysis for the Advancing Out-of-School Learning in Mathematics and Engineering project | Proposes an open-source, maintainable system for detecting human activity in video datasets. |
| Shi, W., et, al. 2016 [43], [40] | Robust head detection in collaborative learning environments using AM-FM representations | Focuses on head detection, attention based detection by classifying where faces look, and group interactions based on attention direction detected. This work uses texture by using AM-FM models. |
| Jatla, V., et, al. 2016 [25, 26] | Image processing methods for coronal hole segmentation, matching, and map classification | Explored image processing models that can be used to detect coronal holes automatically. Here an automated segmentation method has been developed that improves significantly over state of the art models . |

## 2.2 Object detection

The goal of object detection methods is to determine where objects are located in an image and determine the category of the object. In this section, I will provide a summary of common object detection datasets and methods. The most commonly used object detection datasets are listed in the table 2.2. The most popular object detection algorithms are as described in the table 2.3.

Table 2.2: Summary of commonly used object detection datasets. These datasets contain a large number of classes with images extracted from multiple sources. MS-COCO [32] dataset contains a keyboard class, making it of particular interest for this research.

| **Dataset** | **Summary** |
|---|---|
| MS-COCO [32] | • 330k images($> 200K$ labels)<br>• 80 object categories<br>• 5 captions per image<br>• 2,50,000 people with key points, Object segmentation.<br>• 1.5 million object instances |
| ImageNet [10] | • 1,500,000 images with multiple bounding boxes and respective class labels.<br>• Over 500 images per category |
| PASCAL VOC 2007 [18] | • 20 classes, including person, animal, vehicle, and indoor.<br>• Train/validation/test, 9,963 images containing 24,640 annotated objects. |
| PASCAL VOC 2012 [19] | • 20 classes.<br>• The train/val data has 11,530 images containing 27,450 ROI annotated objects and 6,929 segmentations. |

**Faster R-CNN**

Faster R-CNN [39] is the modified version of Fast R-CNN. The major difference is that Fast R-CNN uses the selective search for generating Regions of Interest, while Faster R-CNN uses "Region Proposal Network," RPN. RPN takes image feature

Table 2.3: Summary of the most common object detection methods. These methods are supported officially by deep learning libraries, such as PyTorch, making them easily accessible.

| **Author** | **Method** | **Summary** | **Datasets** | **Results** |
|---|---|---|---|---|
| Redmon,et al. 2018 [38] | YOLOv3 | • Single neural network to full image.<br>• Fast and accurate | • PASCAL VOC<br>• MS COCO [32] | • YOLOv3 runs significantly faster than other detection methods with comparable performance and with high accuracy.<br>• The mAP(mean Average Precision) is 57.9% on COCO test-dev. |
| Shaoqing Ren, et al. 2017 [39] | Faster R-CNN | • End-to-end trained RPN to generate high quality region proposals, which are used by Fast R-CNN for detection | • PASCAL VOC 2007<br>• PASCAL VOC 2012<br>• MSCOCO | • mAP of 73.2% on PASCAL VOC 2007<br>• mAP of 70.4% on PASCAL VOC 2012 |
| Wei Liu, et al.2017 [33] | SSD | • End-to-end CNN passing input image through series of convolutional layers, generating bounding boxes.<br>• Works well with larger input image | • PASCAL VOC<br>• MSCOCO<br>• ILSVRC | • mAP 74.3% on VOC2007 test<br>• On VOC2007 test, operated at 59 FPS with mAP 74.3%, vs. Faster R-CNN 7 FPS with mAP 73.2% or YOLO 45 FPS with mAP 63.4% |

maps as an input that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. RPN is a fully convolutional network that continuously generates object bounds and objectness scores at each position. This is trained end-to-end for high-quality region proposals which are further used by Fast R-CNN for detection. An RoI pooling layer is applied on these proposals to bring down these proposals to the same size. These proposals are passed through the FCN layer, which has softmax and linear regressor at its top to classify and output bounding boxes. The architecture is shown in Fig. 2.1.

### Single Shot Detector

Single Shot Detector (SSD) [33] is a method for detecting objects in images using

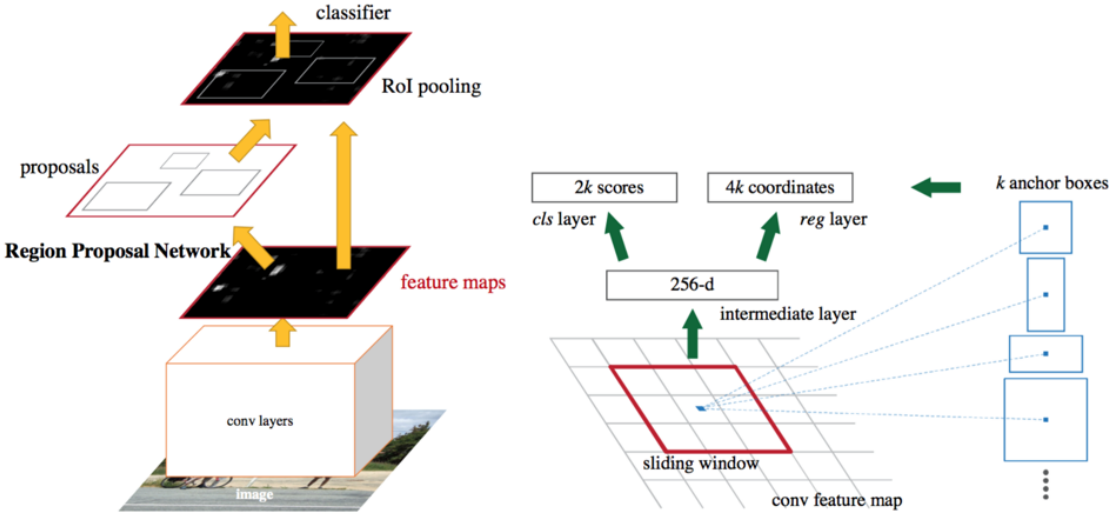



Figure 2.1: Architecture of Faster R-CNN [39].

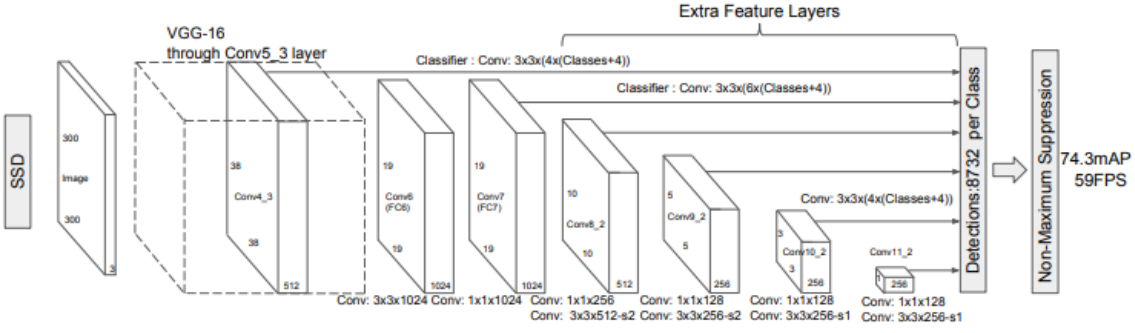



Figure 2.2: Architecture of SSD [33].

a single deep neural network. The SSD approach discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios. After discretizing, the method scales per feature map location. The Single Shot Detector network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes. The architecture is shown in Fig. 2.2.

### YOLO

YOLO [38] shown in Fig. 2.3 uses a single neural network trained end-to-end that takes a photograph as input and predicts bounding boxes and class labels for each

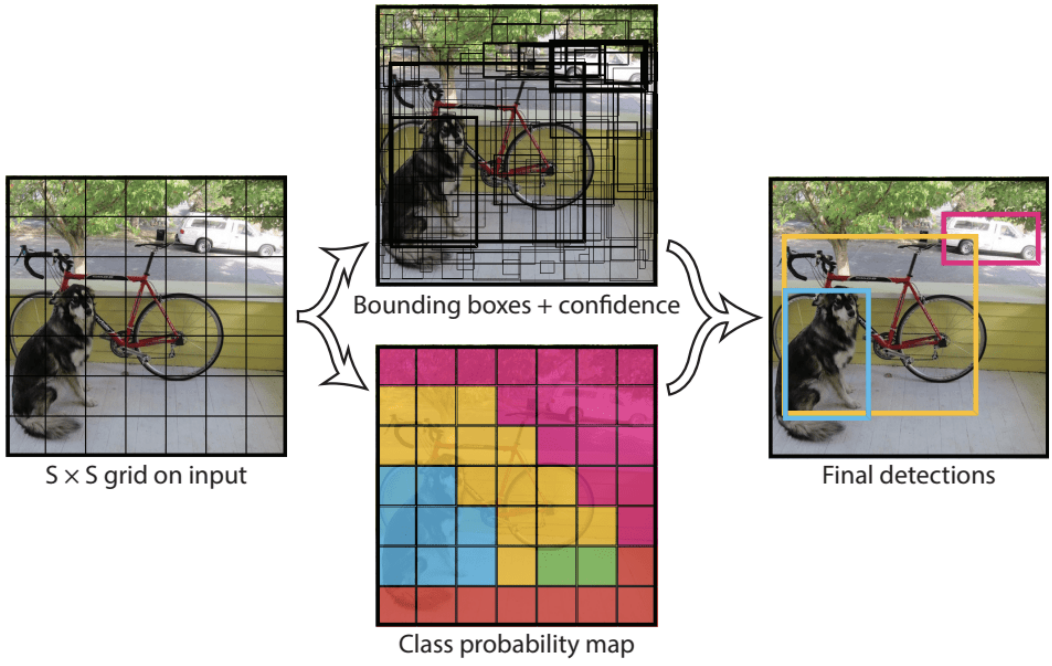



Figure 2.3: YOLO [38].

bounding box directly. This model works by first splitting the input image into a grid of cells, where each cell is responsible for predicting a bounding box if the center of a bounding box falls within the cell. Each grid cell predicts a bounding box involving the x, y coordinate, the width and height, and the confidence. A class prediction is also based on each cell.

### 2.2.1 Object detection libraries

This section briefly describes the frameworks used in the thesis to implement object detection. There are multiple github repositories that provide object detection. In this thesis, we used the frameworks supported by well established companies in AI research, such as Facebook [51] and OpenMMLab [7].

**Detectron2**

Detectron2 [51] is Facebook AI Research's software system that implements state-of-the-art object detection algorithms. Detectron2 is powered by Pytorch deep learning frameworks. It includes features such as Deeplab, Cascade R-CNN, rotated bounding boxes, etc. The training is faster, and models can be exported to Torchscript or Caffe format for deployment.

**MMDetection**

MMDetection [7] is an open-source object detection toolbox based on PyTorch. It is a part of the OpenMMLab project. The toolbox stems from the codebase developed by the MMDet team, who won the COCO Detection Challenge. The detection framework is divided into different components so that one can easily use a customized object detection framework by combining different modules. It supports multiple state-of-the-art frameworks like Faster R-CNN, YOLO, SSD, Mask R-CNN etc. Once the data is arranged in a particular COCO format, multiple frameworks can be trained by just changing configuration files. All the operations are run on GPUs, so the training speed is comparable to other popular networks like Detectron2 and Tensorflow.

## 2.3 Object tracking

Object tracking methods aim to determine the spatial coordinates of an object (initialized previously) in future frames. This section summarizes common object tracking datasets and trackers supported in OpenCV and state-of-the-art algorithms for tracking. The most commonly used datasets for object tracking are listed in table 2.4. OpenCV object tracking API was introduced in OpenCV 3.0. A total of 8 tracking algorithms are available in OpenCV. They are BOOSTING, MIL, KCF, TLD, MEDIANFLOW, GOTURN, MOSSE, and CSRT. Generally, tracking algorithms are faster than detection; the reason is that the algorithm already knows the object's

appearance. All the tracking algorithms are summarized in table 2.5. In addition, the state-of-the-art tracking algorithms are summarized in table 2.6.

Table 2.4: Summary of object tracking datasets used to train and test single-target and multi-target tracking.

| **Dataset** | **Summary** |
|---|---|
| MOT17 [35] | • Mutli-target tracking.<br>• 42 sequences with crowded scenarios, camera motions, and weather conditions. |
| MOT20 [9] | • 8 new sequence depicting very crowded challenging scenes. |
| TrackingNet [37] | • A Large-Scale Dataset and Benchmark for Object Tracking in the Wild.<br>• $> 30K$ Video Sequences.<br>• $> 14M$ Bounding Boxes.<br>• Diversity ensured by YouTube |
| VOT2018 [31] | • 235 sequences, carefully selected to obtain a dataset with long sequences containing many target disappearances from LTB35.<br>• Twenty sequences were obtained from the UAVL20 [36], six sequences were taken from Youtube, six sequences were generated from the omnidirectional view generator AMP [96] to ensure many target disappearances.<br>• Sequence resolutions range between 1280×720 and 290×217. The dataset contains 14687 frames, with 433 target disappearances.<br>• Each sequence contains on average 12 long-term target disappearances, each lasting on average 40 frames. |

**Boosting Tracker [20]:**

This tracker is based on AdaBoost's online edition. There is no special reason to use these trackers since there are more advanced trackers such as KCF, MIL. The tracking performance is mediocre and does not reliably track failure. It is not real time. It fails with random and fast movements.

**MIL Tracker [2]:**

The perfomance is better compared to BOOSTING, since it does not drift much. It

does good job under partial occlusion. It fails with random and fast movements. It is not real time.

**KCF Tracker [22]:**

Accuracy and speed are both better than previous two trackers. It reports tracking failure. It does not recover from full occlusion and fast movements. It is real time. This is the best peforming algorithm that was used in this Thesis.

**TLD Tracker [28]:**

This tracker decomposes the task into (short term) tracking, learning, and detection. The tracker follows the object from frame to frame. The detector localizes all appearances that have been observed so far and corrects the tracker if necessary. It works under occlusion. It is not real time. It fails with fast and random movements.

**MEDIANFLOW Tracker [29]:**

This tracker tracks the object both in forward and backward direction. It reliably reports tracking failure. It works well when there is no occlusion. It fails under large motions. It is real time.

**GOTURN Tracker [21]:**

Out of all the trackers, this is the only one based on Convolutional Neural Networks. It does track pretty well when the object is in training set. The tracker has a hard time tracking part of an object. It is not real time.

**MOSSE Tracker [4]:**

It is the fastest of all the available trackers. It is easy to implement. It fails with random movements. But due to its speed, on a performance scale it lags behind.

**CSRT Tracker [34]:**

It uses the spatial reliability map for adjusting the filter support to the part of the selected region from the frame for tracking. It is not real time.

Table 2.5: Summary of object tracking methods available in OpenCV. These are typically fast methods that do not use deep learning.

| **Method** | **Summary** |
|---|---|
| Boosting[20] | • Old and Performance is mediocre<br>• Does not know when tracking is failed. |
| Multiple Instance Learning [2] | • Performance is pretty good, performs well under partial occlusion.<br>• Tracking failure is not reported reliably.<br>• Does not recover from full occlusion. |
| Kernelized Correlation Filters [22] | • Accuracy and Speed are both better than MIL.<br>• Reports tracking Failure better than MIL and BOOSTING.<br>• Does not recover from full occlusion |
| Tracking, Learning and Detection [28] | • Best under occlusion over multiple frames and over scale changes.<br>• Lots of false position. |
| Median Flow [29] | • Excellent tracking failure reporting. Works very well when the motion is predictable and there is no occlusion.<br>• Fails under large motion. |
| GO TURN [21] | • Fails on tracking objects that are not in the training set.<br>• Hard time tracking part of the object. |
| MOSSE [4] | • Operates at higher fps (450 and even more).<br>• Easy to implement and accurate |
| DCF-CSR [34] | • Operates at 25fps<br>• Gives High Accuracy |

## 2.4 Uniqueness of AOLME Dataset

AOLME is different from other typical datasets in the way that it primarily includes long videos which are over an hour. Each video has multiple activities but not limited to typing, talking, eating, and writing. Many challenges impact the results like occlusion, multiple camera angles, illumination issues, multiple people performing the same activity, fast and random movements, people moving across the videos,

Table 2.6: Summary of State-of-the-Art Object tracking algorithms.

| **Author** | **Method** | **Summary** | **Datasets** |
|---|---|---|---|
| Qiang Wang,et al. 2019 [50] | Fast Online Object Tracking and Segmentation, A Unifying Approach SiamMask | • 3rd best model for Visual Object Tracking on YouTube-VOS<br>• Single Object Tracking<br>• Produces segmentation masks and bounding boxes at 55fps<br>• Real time and fastest | VOT-2016, VOT-2018, DAVIS-2016, DAVIS-2017 |
| Yifu Zhang , et al. 2020 [52] | A Simple Baseline for Multi-Object Tracking | • SOTA for Multi-Object Tracking on MOT16<br>• First among all online trackers<br>• Operates at 30fps<br>• Anchor-free object detection to reduce ambiguity, parallel branch for Re-ID features | 2DMOT15, MOT16, MOT17, MOT20 |

and activities in the background.

# Chapter 3

# Dataset

The Advancing Out-of-school Learning in Mathematics and Engineering (AOLME) project is an after-school program collaboratively implemented by the Department of Electrical and Computer Engineering and the Department of Language, Literacy, and Sociocultural Studies. AOLME generated a large amount of multimedia data that includes around 2200 hours of group interactions, monitor data, and screen recordings.

A group in the AOLME setting has three to five students, one facilitator, and a co-facilitator. The interactions between the group were recorded using a video camera. The videos have $1920 \times 1080$ resolution and 30 or 60 frames per second.

In this chapter, we give a detailed description of AOLME data terminology and organization. Following this, we describe the ground truth generation to train object detectors for detecting hands and keyboards in group interactions.

## 3.1 Organization of Dataset

Fig. 3.1 shows the organization of the AOLME Dataset. Group video data is collected across three years which are named Cohorts. Each Cohort is again organized into levels based on Spring, Summer, and Fall. Each Level again has two schools: Rural and Urban.

For the dataset, we have:

- Each school has 3 to 7 groups.
- Each group has 3 to 5 members.
- Typically, per level a group does 10 to 12 sessions.
- A session lasts anywhere between 45 minutes to 90 minutes.

  The videos are organized using the following naming convention:

  - **C1L1P-A** : Cohort 1, Level 1, Rural, Group A (Includes all sessions from Group)
  - **C1L1P-A, Mar02** : Cohort 1, Level 1, Rural, Group A (Single Session)

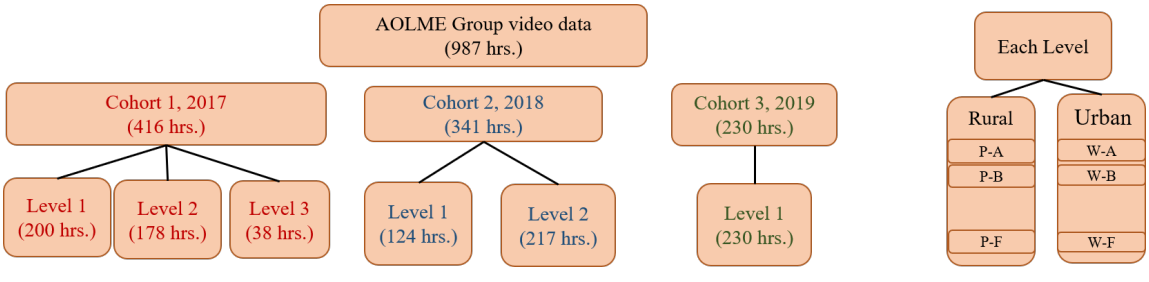



Figure 3.1: AOLME Dataset Organization.

## 3.2 Ground Truth

### 3.2.1 Generation of Ground Truth

To generate ground truth, MATLAB Video Labeler 2018Rb was used. Each video was reviewed in three second segments. Within each segment, a spatiotemporal bounding box was used to mark each activity. The list of activities included: typing/no-typing and writing/no-writing. For each activitiy, we also stored the anonymized student, the start time, and activity duration.

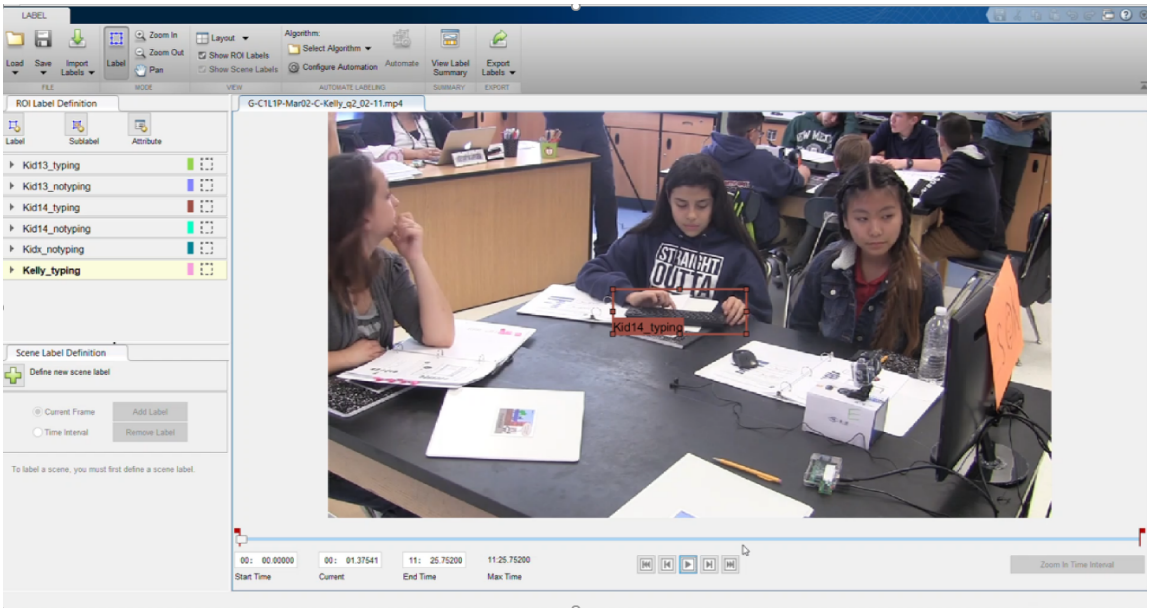



Figure 3.2: Ground truth labeling process.

The ground truth datasets are summarized in Tables 3.1 and 3.2. A total of 83 hours and 49 hours of video data was analyzed for labeling typing and writing activities, respectively.

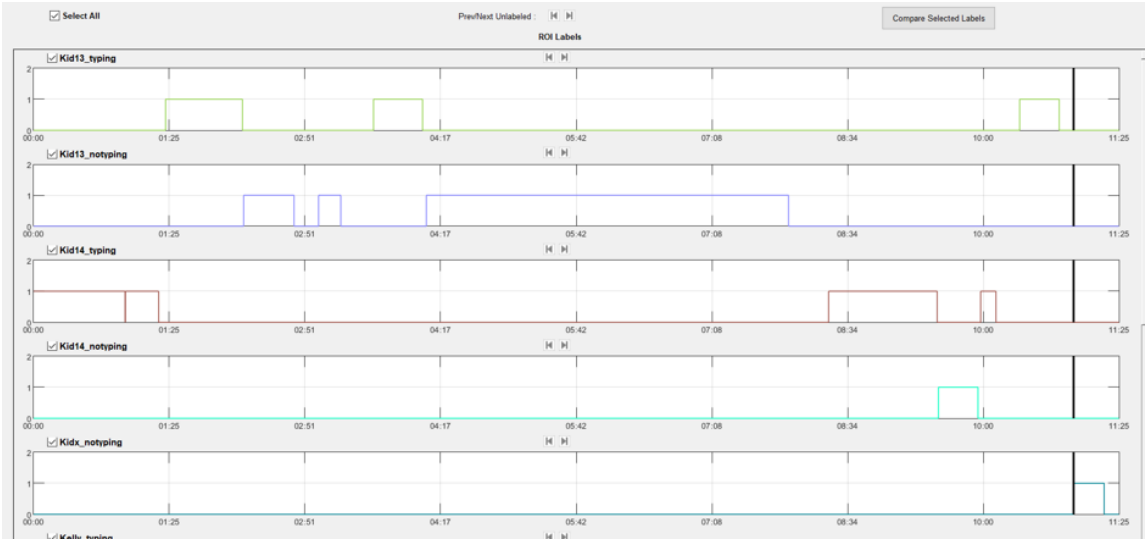



Figure 3.3: Visualizing ground truth using video labeler tool.

### 3.2.2 Testing Dataset

All the sessions handpicked by the College of Education are used for testing. These include 13 different sessions from Urban and Rural schools, which include 37 students and 10 facilitators. The total length of these videos is over 21 hours. The dataset is summarized in Table 3.3.

Table 3.3: AOLME Testing dataset.

| Cohort | Level | Date | Group | School |
|---|---|---|---|---|
| 1 | 1 | Mar-02 | B | Rural |
| 1 | 1 | Mar-30 | C | Rural |
| 1 | 1 | Apr-06 | C | Rural |
| 1 | 1 | Apr-13 | C | Rural |
| 1 | 1 | Mar-02 | E | Rural |
| 2 | 1 | Feb-23 | B | Rural |
| 2 | 1 | Apr-12 | C | Rural |
| 2 | 1 | Mar-08 | D | Rural |
| 2 | 1 | Apr-12 | E | Rural |
| 2 | 1 | Feb-27 | B | Urban |
| 3 | 1 | Apr-11 | C | Rural |
| 3 | 1 | Feb-21 | D | Rural |
| 3 | 1 | Mar-19 | D | Urban |

Table 3.1: Typing and no typing ground truth on AOLME group videos. These videos are of 848x480 resolution. Playback time typically ranges from 11 to 16 minutes at 30 or 60 FPS. We use boldface to identify video sessions where ground truth was provided over the entire session.

**Cohort 1, Level 1**

**Rural**

| Group | Dates | # videos proc. | Total Videos | Hours proc. | Total hours | Typing | No Typing |
|---|---|---|---|---|---|---|---|
| A | Feb 16, Feb 25, Mar 02, Mar 09 Apr 06, Apr 13, Apr 20 | 7 | 59 | 1.65 | 17.79 | 0.26 | 0.63 |
| B | Mar 02, Mar 09, Mar 30, Apr 06 Apr 27, May 04, May 06, May 11 | 17 | 61 | 3.69 | 14.13 | 0.45 | 1.03 |
| C | Feb 16, Feb 25, Mar 02, Mar 09 **Mar 30, Apr 13**, Apr 20, **May 04** | 35 | 68 | 8.13 | 14.05 | 1.24 | 1.32 |
| D | Mar 09, Apr 06, Apr 13, Apr 20 | 9 | 14 | 3.77 | 6.42 | 0.5 | 0.04 |
| E | Feb 25, **Mar 02** | 9 | 23 | 1.61 | 4.01 | 0.4 | 0.67 |

**Urban**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A | Feb 21, Feb 28, Mar 07, Mar 28 Apr 25 | 6 | 36 | 1.3 | 7.05 | 0.4 | 0.45 |
| B | May 06 | 1 | 14 | 0.27 | 2.89 | 0.11 | 0.03 |
| C | Feb21 | 1 | 9 | 0.19 | 1.38 | 0.04 | 0.11 |
| D | Feb28 | 1 | 8 | 0.15 | 1.48 | 0.05 | 0.03 |
| **Total** | | 35 | 292 | 20.76 | 69.2 | 3.45 | 4.31 |

**Cohort 2, Level 1**

**Rural**

| Group | Dates | # videos proc. | Total Videos | Hours proc. | Total hours | Typing | No Typing |
|---|---|---|---|---|---|---|---|
| B | **Feb 23** | 6 | 6 | 1.81 | 1.80 | 0.21 | 1.37 |
| C | **Apr 12** | 6 | 6 | 1.94 | 1.94 | 0.01 | 1.18 |
| D | **Mar 08** | 5 | 5 | 1.78 | 1.78 | 0.31 | 1.08 |

**Urban**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A | Apr 10 | 2 | 5 | 0.79 | 1.54 | 0.02 | 0.0 |
| B | **Feb 27** | 4 | 4 | 1.39 | 1.39 | 0.41 | 0.83 |
| **Total** | | 23 | 26 | 7.7 | 8.44 | 0.959 | 3.45 |

**Cohort 3, Level 1**

**Rural**

| Group | Dates | # videos proc. | Total Videos | Hours proc. | Total hours | Typing | No Typing |
|---|---|---|---|---|---|---|---|
| C | **Apr 11** | 5 | 5 | 1.78 | 1.78 | 0.31 | 1.08 |
| D | **Feb 21** | 5 | 5 | 1.77 | 1.77 | 0.15 | 1.08 |

**Urban**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A | **Mar 19** | 4 | 4 | 1.35 | 1.35 | 0.18 | 1.11 |
| **Total** | | 14 | 14 | 4.9 | 4.9 | 0.64 | 3.34 |

Table 3.2: Writing and no writing ground truth on AOLME group videos. These videos are of 848x480 resolution. Playback time typically ranges from 11 to 16 minutes at 30 or 60 FPS. We use boldface to identify video sessions where ground truth was provided over the entire session.

**Cohort 1, Level 1**

**Rural**

| Group | Dates | # videos proc. | Total Videos | Hours proc. | Total hours | Writing | No Writing |
|---|---|---|---|---|---|---|---|
| B | **Mar 02** | 9 | 9 | 1.5 | 1.5 | 0.5 | 2.85 |
| C | Feb 16, Feb 25, Mar 09, **Mar30** Apr 06, **Apr 13**, Apr 20,May04, May11 | 38 | 69 | 8.86 | 15.07 | 2.22 | 7.69 |
| D | Mar 02, Mar09, Mar30, Apr 06, Apr 13 Apr 20 | 18 | 29 | 7.03 | 9.45 | 1.08 | 0 |
| E | **Mar 02** | 8 | 8 | 1.42 | 1.42 | 0.87 | 3.61 |

**Urban**

| Group | Dates | # videos proc. | Total Videos | Hours proc. | Total hours | Writing | No Writing |
|---|---|---|---|---|---|---|---|
| A | Feb 14, Feb 21, Feb 28, Apr 04 | 13 | 30 | 2.33 | 5.49 | 0.56 | 0 |
| **Total** | | 86 | 149 | 21.12 | 32.93 | 5.23 | 14.15 |

**Cohort 2, Level 1**

**Rural**

| Group | Dates | # videos proc. | Total Videos | Hours proc. | Total hours | Writing | No Writing |
|---|---|---|---|---|---|---|---|
| B | **Feb 23** | 6 | 6 | 1.80 | 1.80 | 0.15 | 2.83 |
| C | **Apr 12** | 6 | 6 | 1.95 | 1.95 | 0.97 | 1.23 |
| D | **Mar 08** | 5 | 5 | 1.78 | 1.78 | 0.88 | 0.01 |
| E | **Apr 12** | 6 | 6 | 1.85 | 1.85 | 0.33 | 2.16 |

**Urban**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| A | Feb 20, Apr 10 | 3 | 9 | 1.09 | 2.57 | 0.34 | 0.0 |
| **Total** | | 26 | 32 | 8.47 | 9.95 | 2.67 | 6.23 |

**Cohort 3, Level 1**

**Rural**

| Group | Dates | # videos proc. | Total Videos | Hours proc. | Total hours | Writing | No Writing |
|---|---|---|---|---|---|---|---|
| C | **Apr 11** | 5 | 5 | 1.78 | 1.78 | 0.84 | 2.74 |
| D | Feb14, **Feb21** | 6 | 9 | 1.78 | 2.94 | 0.14 | 1.15 |

**Urban**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| D | **Mar19** | 4 | 4 | 1.35 | 1.35 | 0.16 | 0.19 |
| **Total** | | 15 | 18 | 4.91 | 6.07 | 1.14 | 4.08 |


(a) (b) (c) (d) (e) (f)

Figure 3.4: Some of the images representing test set.

## 3.3 Datasets for keyboard and hand detection

### 3.3.1 Dataset for keyboard detection

Two Frames and corresponding bounding boxes are extracted for every minute from typing-no typing ground truth and exported as CSV files. Each image is of size 858×480 pixels. Train and Test splits are done following the standard of different sessions. The dataset used for keyboard detection is as shown in the table 3.4

Table 3.4: Dataset for keyboard detection.

| | **No. of Groups** | **No. of Sessions** | **No. of Images** |
|---|---|---|---|
| **Train** | 9 | 33 | 700 |
| **Validation** | 4 | 4 | 100 |
| **Test** | 6 | 7 | 648 |

### 3.3.2 Dataset for hand detection

Here as writing ground truth does not include all hands in the image, makesense.ai (Free open-source and Online labeling tool) [44] is used to generate ground truth. A total of 718 images were used here, with all the hands marked as shown in Fig. 3.5. The dataset used for hand detection is shown in table 3.5.

Table 3.5: Dataset for hand detection.

| | **No. of Groups** | **No. of Sessions** | **No. of Images** | **No. of Hand Instances** |
|---|---|---|---|---|
| **Train** | 9 | 33 | 305 | 1803 |
| **Validation** | 4 | 4 | 100 | 714 |
| **Test** | 6 | 7 | 313 | 2031 |

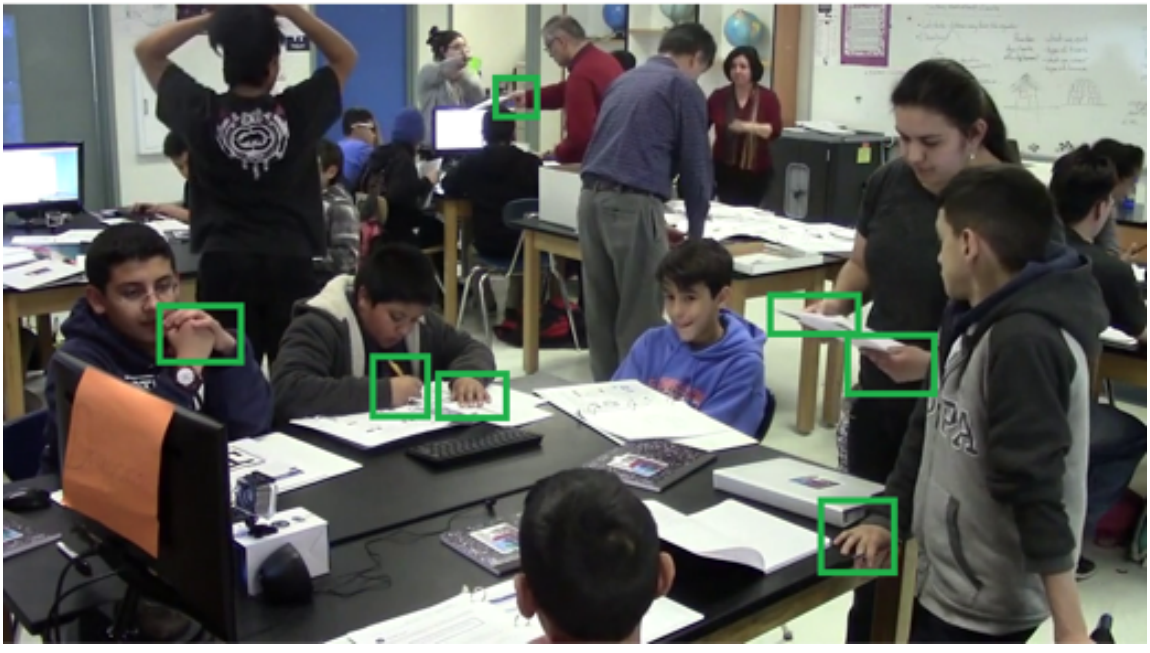


Figure 3.5: Sample from hand detection dataset.

### 3.3.3 Testing Dataset

Training, Validation, and Testing are mutually exclusive with respect to sessions. Throughout all the experiments, the dataset used for testing is from sessions listed in the table 3.6.

Table 3.6: Dataset for Testing.

| **Group Video** | **Date** |
|---|---|
| C1L1P-C | Mar 30 5 |
| C1L1P-C | Apr 13 5 |
| C1L1P-E | Mar 02 |
| C2L1P-B | Feb 23 |
| C2L1P-D | Mar 08 |
| C3L1P-C | Apr 11 |
| C3L1P-D | Mar 19 |

# Chapter 4

# Methodology

This chapter provides summary of methods for keyboard detection and tracking, hand detection and optimal data augmentation parameters.

## 4.1 Keyboard detection and tracking

A top-level diagram for the keyboard detection and tracking system is shown in Figure 4.1. Generally, tracking algorithms are faster than detecting an object on every video frame. During tracking, the search region is restricted for reasonable motions. The thesis considers an adaptive system where object detection is performed once every n frames, followed by object tracking in the remaining frames.

I implemented a combined network of detection and fast-tracking such that the accuracy is the same or better than detection alone while being significantly faster. The fast-tracking is achieved by combining the fast tracker, KCF, selected as the fastest and best performing tracking methods implemented in OpenCV.

A session video is given as input, and the object of interest is detected for the first frame, and it is tracked for the next 5-second interval. Then, it is again re-initialized

with the detection after every 5-second interval. This process is repeated until the end of the video. Fig. 4.2 shows an example from keyboard detection and Fig. 4.3 shows an example from tracking.

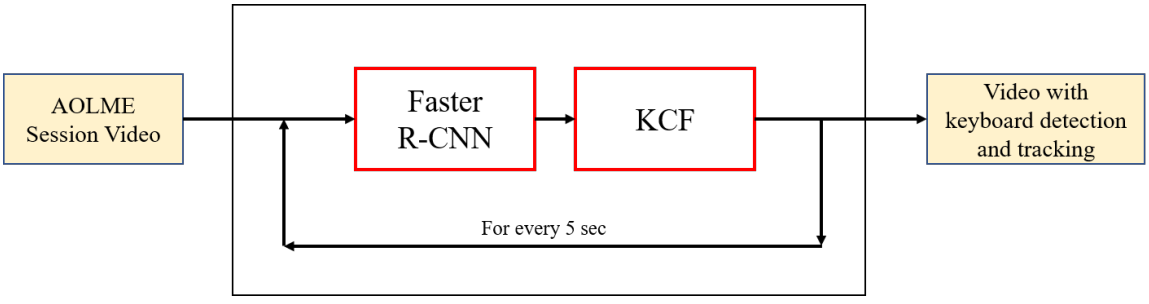



Figure 4.1: System for keyboard detection and tracking.

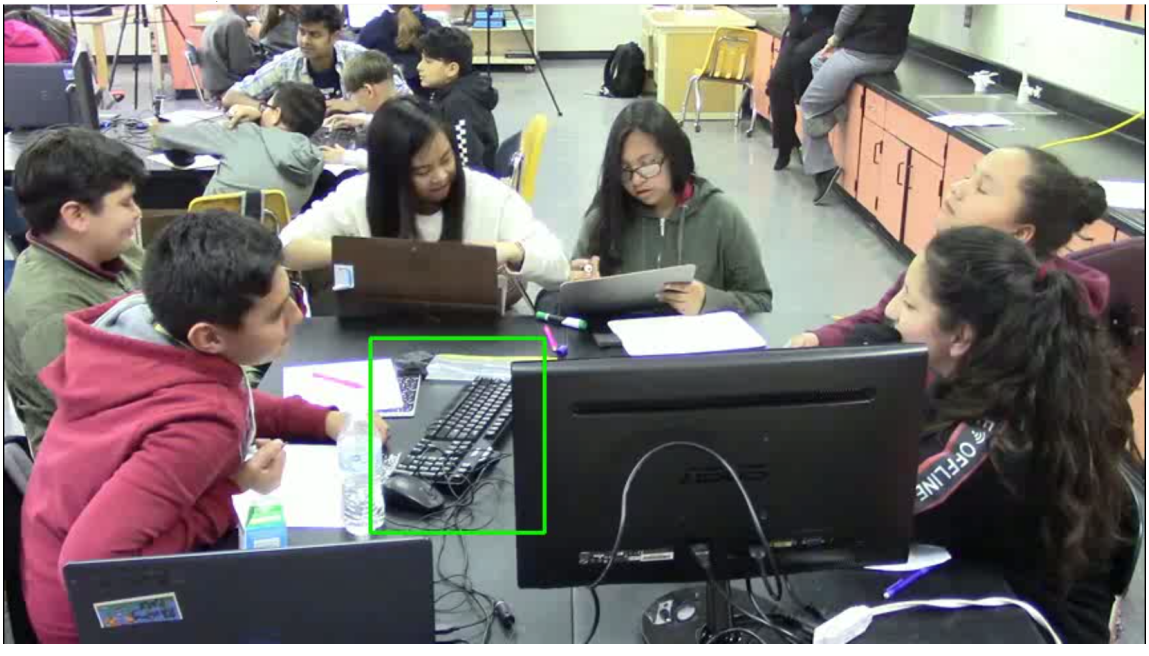


Figure 4.2: An example of keyboard detection.


Figure 4.3: An example of keyboard tracking.

## 4.2 Top-level diagram for hand regions

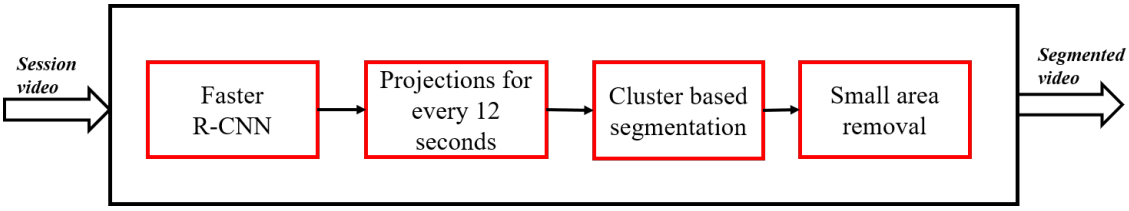
The top-level diagram and corresponding pseudo code for generating hand regions is as shown in Fig. 4.4. It has five different steps. It takes a session video as input and produces the output video with bounding boxes around potential instances of activities involving hand movement.

**Components of Top-level diagram for hand regions**:

1. **Object Detection**

   Input Video is passed through an Object Detector (Faster-RCNN) to detect hands at the rate of one frame per second. The output of the hand detection method is assumed to be 1 over pixel regions that represent hand regions, and 0 over other regions. It is shown in Fig. 4.5.

2. **Projection for every 12 seconds**




**function** DETECTHANDS($w_*$, V, $\mathtt{a_{th}}$)
▷ **Input**:
▷ $w_*$ represents a pre-trained single-frame hand detector.
▷ V represents a short Video segment of fixed n seconds duration.
▷ $\mathtt{a_{th}}$ represents a minimum area requirement.
▷ **Output**:
▷ H contains the detected hand regions for each 12-second video segment.
BI $\leftarrow w_*$(V) ▷ detect hands at the rate of one frame per second.
H $\leftarrow \{\}$ ▷ initialize H to store hand detections.
**for** each 12-second video segment i: **do**
Project the detected hand regions using:
$\mathtt{PI_i} \leftarrow \sum_s \mathtt{BI_s}$
Cluster the projected hand regions using:
$\mathtt{CI_i} \leftarrow$ **Cluster**($\mathtt{PI_i}$)
Remove small hand regions of far-away groups:
$\mathtt{H_i} \leftarrow$ **AreaThreshold**($\mathtt{CI_i}$, $\mathtt{a_{th}}$)
H $\leftarrow$ **Append**(H, $\mathtt{H_i}$)
**end for**
**return** H
**end function**

Figure 4.4: Proposed hand detection method using time-projections, clustering, and small region removal

All the detections for every 12 seconds are added up and projected on to an image as shown in Fig. 4.6. The projected images $\{PI_1, PI_2, ..., PI_{n/12}\}$ can hold a maximum of 12 that represents hand detection over all images, and a minimum of 0 that represents the lack of any hands detected over any image.

3. **Cluster based segmentation**

   To account for occlusion, appearence, and disappearance, we apply a clustering method over the projected image. ISODATA [3] thresholding is used to auto-

Figure 4.5: Video frame showing hands detected.

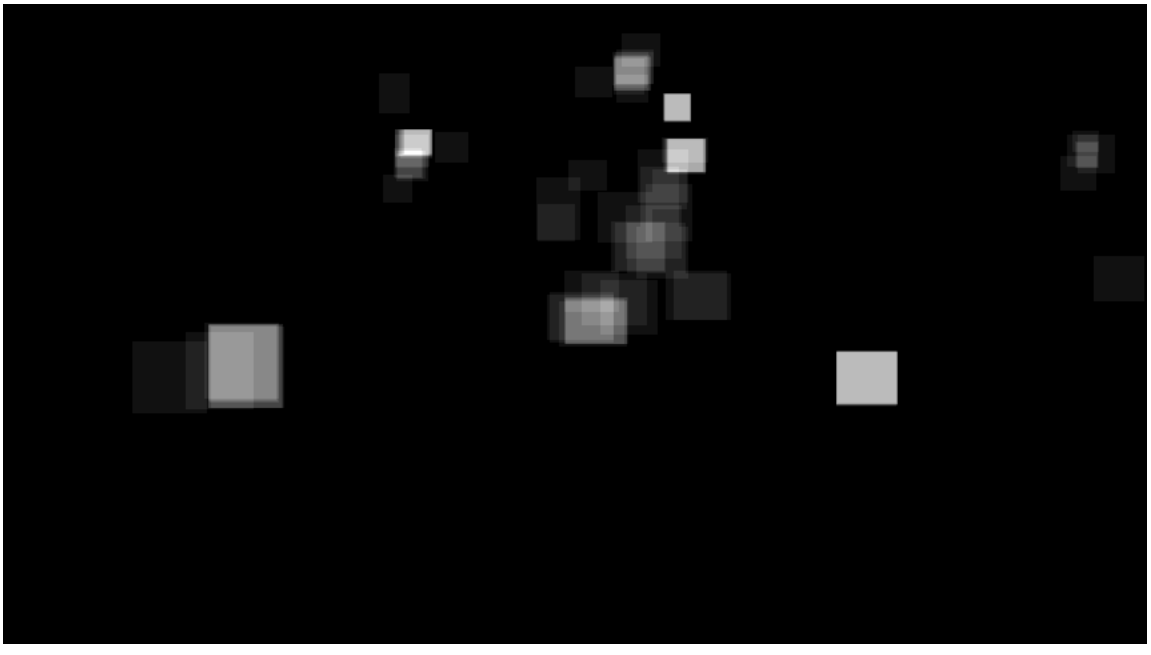


Figure 4.6: Video frame showing projected detections for 12-seconds interval

matically find a threshold value for a given grayscale image. ISODATA is an

unsupervised, iterative process for computing the minimum Euclidean distance when assigning each candidate cell to a cluster.

A variety of clustering techniques i.e., Isodata, Li, Mean, Minimum, Otsu, Triangle, Yen [49] as shown in figure 4.7 were tried. Over an exploratory dataset, ISODATA gave the best over-segmentation results by capturing the writing regions using the minimum number of clusters. It is as shown in figure 4.8.

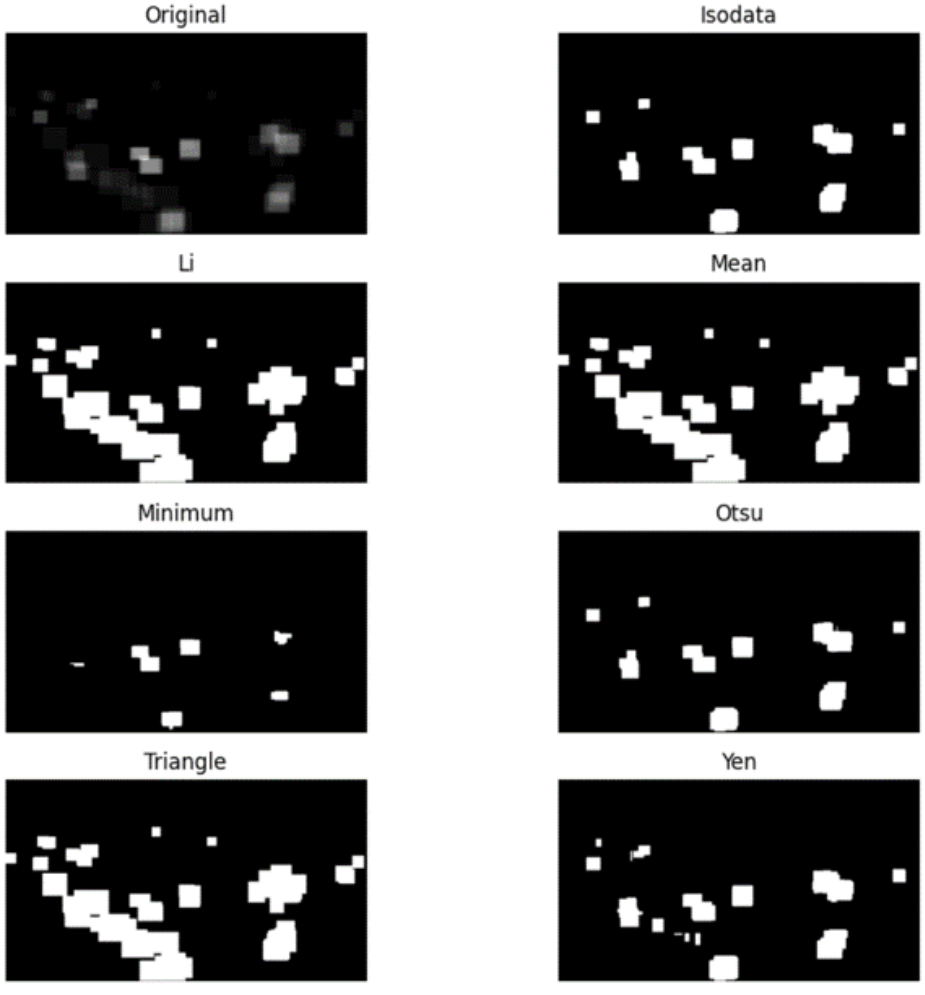



Figure 4.7: picture showing different clustering techniques.

4. **Small area removal**

Following this, the hand clusters that correspond to distant groups are rejected

Figure 4.8: Video frame showing isodata clustering.

based on a cluster area constraint as shown in Fig. shown in Fig. 4.9.

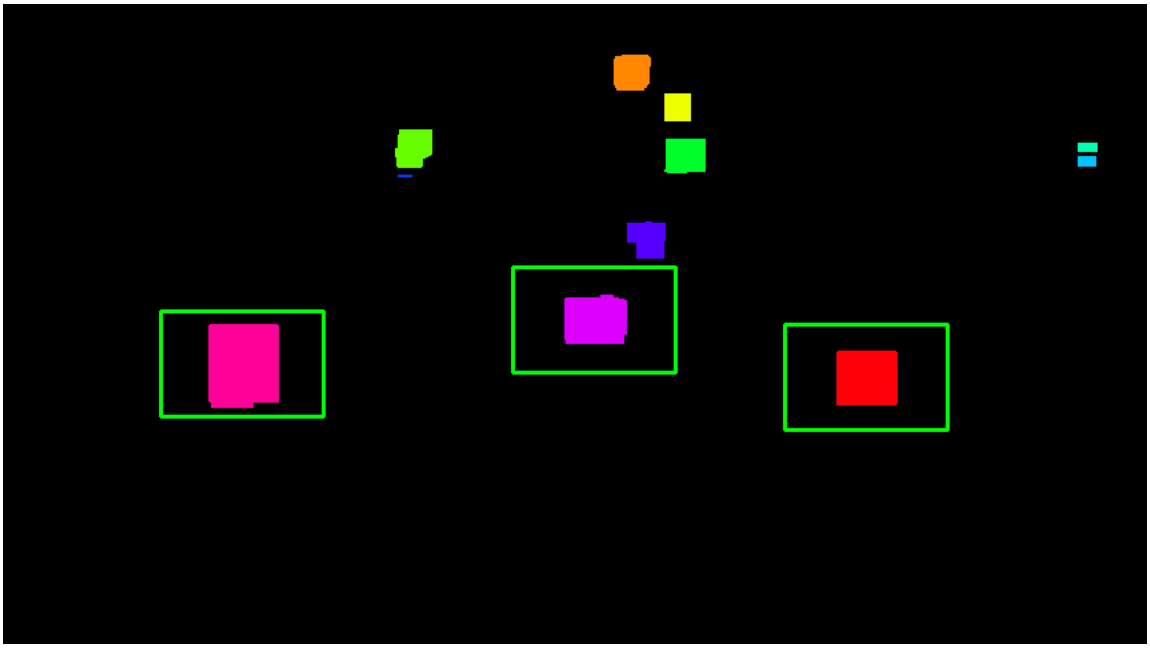


Figure 4.9: Video frame showing removal of smaller regions.

5. **Segmented video**

   Bounding boxes are drawn around remaining regions with the median width and height from ground truth regions as shown in Fig. 4.10.

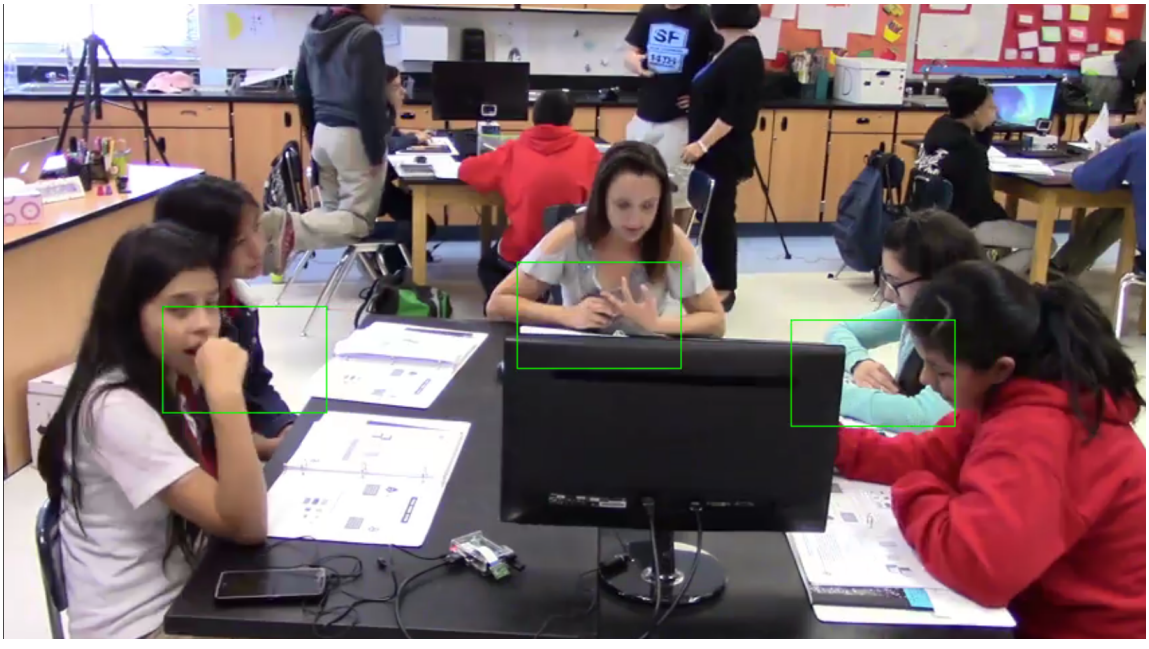


Figure 4.10: Video frame with bounding boxes around hand activity regions.

## 4.3 Optimal data augmentation study

A data augmentation study was performed here to improve hand detection results. First, an optimal range of angles for shearing, rotation, and pixels to be translated are determined separably. Once all the optimal values are found, an optimal study of probabilities to apply these augmentations is performed. Once all the optimal values are found, finally, the model is trained with optimal data augmentation parameters.

### 4.3.1 Experimental Setup

To perform all these experiments, the library used is MMDetection [7]. The detection algorithm used is Faster R-CNN. The study was done on RTX 5000 GPU with a learning rate of 0.001. The number of epochs used here is 12, as recommended. The mini-batch size is two images with two workers.

**Randomized affine transformations**

**Affine transformation.** Transformation of an image such that parallel lines in an image remain parallel after the transformation. Scaling, translation, rotation shearing are all examples of affine transformations.

**Rotate angle:**

The image and the corresponding bounding box are rotated with the angle $\theta$. A Positive angle rotates it counter clockwise while the negative angle rotates the image clockwise.

**Translation pixels:**

The image and the corresponding bounding box are translated horizontally towards left and right based on the values.

**Shear angle:**

Shearing slides one edge of an image along the X or Y axis, creating a parallelogram. Horizontal shear slides an edge along the X direction and the corresponding bounding box too. Similarly, vertical shear slides an edge along the Y direction. Shear angle specifies the number of degrees to shear the images. In the experiments, horizontal shearing is used.

The optimal values for shear angle, rotate angle and translation pixels are obtained using the pseudo code shown in Fig. 4.11. For Shear Angle optimization, the values of $\Theta$ are $1^\circ$, $2^\circ$, $4^\circ$, $8^\circ$, $16^\circ$, $32^\circ$. For Rotate Angle Optimization, values of $\Theta$

**for** each p in list(P) **do**
**for** image in images with p **do**
Apply random horizontal flips with p.
Apply scaling of [0.8,1.2] with p.
Apply random shear angle transforms between optimal range with p.
Apply random rotate angle transforms between optimal range with p.
Apply random horizontal translate pixel transforms between optimal range with p.
**end for**
Train the model with this dataset.
Test on validation set and record validation accuracies.
**end for**
***Note*** the p which gave best validation accuracy.

Figure 4.12: Pseudo code for finding optimal probability.

are 1°, 2°, 4°, 8°, 16°, 32°. For translation pixels, values of $\Theta$ are 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 800.

**for** each $\theta$ in list($\Theta$) **do**
Augment all the training images with the $\theta$.
Train the model with the augmented dataset.
Test on validation set.
Record validation accuracy.
**end for**
***Repeat*** the above procedure untill validation accuracy starts decreasing.
***Repeat*** the same procedure for all the transformations, shear, rotate and translation.
***List*** all the optimal ranges for each of the transformation.

Figure 4.11: Pseudo code for finding optimal ranges for each of the affine transform.

Once the optimal ranges are determined separably, an optimal probability was determined using the pseudo code shown in 4.12.

After obtaining optimal ranges for affine transforms and optimal probability, the model is trained with these parameters. So the optimization is performed for probability, p, and range of $\theta$.

# Chapter 5

# Results

This chapter summarizes detection, tracking and projection results. We first present keyboard detection and tracking, followed by hand detection and projections.

## 5.1 Keyboard Detection and Tracking Results

This section describes the results of keyboard detection, tracking and the combined system of detection and tracking.

### 5.1.1 Keyboard detection results

For keyboard detection, library named Detectron2 [51] is used. Detectron2 is Facebook AI Research's next-generation library that provides state-of-the-art detection and segmentation algorithms. The method used for keyboard detection is Faster R-CNN, which is pre-trained on COCO [32] detection dataset. The re-training time for 300 iterations is about 20 minutes. We were able to achieve a very high AP of 0.92 for keyboard detection. However, the drawback with detection alone is that

the inference time for video is more. Results for keyboard detection are shown in table 5.1. An AP of 0.92 was achieved at an IOU of 0.5. AP @IoU=0.5 represents the model has used threshold of 0.5 to remove unnecessary boxes. Similarly, range 0.50:0.95 represents AP and AR are averaged over multiple IoU values. Specifically, we use 10 IoU thresholds of .50:.05:.95. Area = small represents small objects for area $< 32^2$. AR @ [maxDets=1] and [maxDets=10] mean the maximum recall given 1 detection per image and 10 detections per image correspondingly. Some of the success and failure cases are shown in Fig. 5.1 and 5.2, respectively. The failure cases mainly show the misclassification of the book as a keyboard.

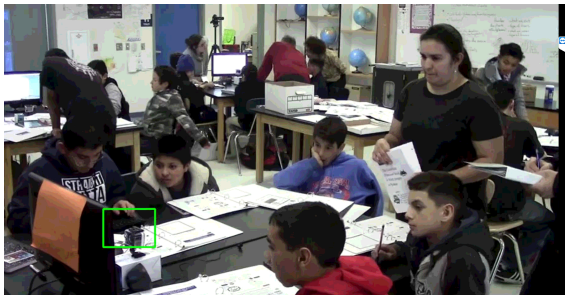
Table 5.1: Average precision (AP) and Average recall (AR) at different IoU ratios for keyboard detection. We are able to achieve a very high AP (0.92) for keyboard detection.

| **Metric** | **Value** |
|---|---|
| AP @[IoU=0.50:0.95 —area = all — maxDets = 100] | 0.614 |
| AP @[IoU=0.50 —area = all — maxDets = 100] | **0.922** |
| AP @[IoU=0.75 —area = all — maxDets = 100] | 0.708 |
| AP @[IoU=0.50:0.95 —area = small — maxDets = 100] | -1.000 |
| AP @[IoU=0.50:0.95 —area = medium — maxDets = 100] | 0.611 |
| AP @[IoU=0.50:0.95 —area = large — maxDets = 100] | 0.620 |
| AR @[IoU=0.50:0.95 —area = all — maxDets = 1] | 0.659 |
| AR @[IoU=0.50 —area = all — maxDets = 10] | 0.676 |
| AR @[IoU=0.75 —area = all — maxDets = 100] | 0.676 |
| AR @[IoU=0.50:0.95 —area = small — maxDets = 100] | -1.000 |
| AR @[IoU=0.50:0.95 —area = medium — maxDets = 100] | 0.670 |
| AR @[IoU=0.50:0.95 — area = large — maxDets = 100] | 0.679 |

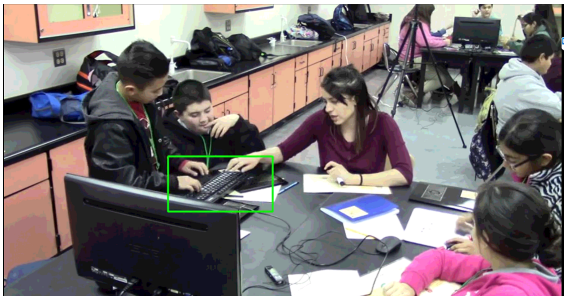
### 5.1.2 Tracking results

To determine how well the trackers perform with AOLME video, I experimented with calculating accuracy and speeds, respectively. To conduct the experiments, the system used is Intel Xeon CPU ES-2640, 16 cores per node at 2.6 GHz with RAM of 64 GB and GPU, Nvidia Tesla K40M. The video I used is of 23.45 minutes @

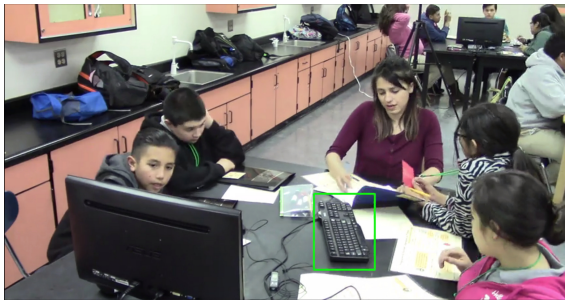
(a) Successful detection with partial occlusion.



(b) Successful detection with partial occlusion.



(c) Successful detection with hands.



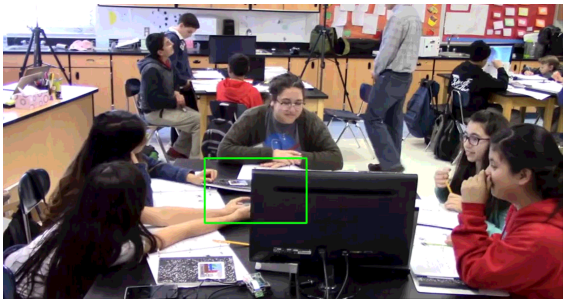
(d) Successful detection without hands.

Figure 5.1: Successful keyboard detections.

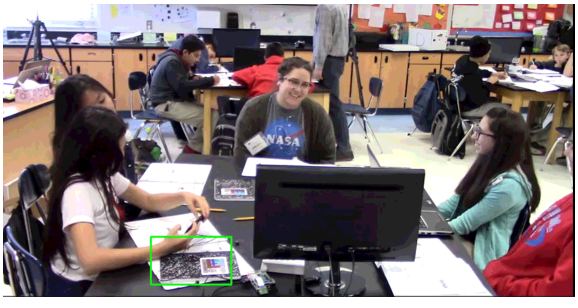
30fps with a resolution of 858×480. All the trackers are initialized with the same object and tracked throughout the end of the video. All the trackers failed with fast movements of the object. However, MOSSE, KCF, and Median flow proved to be real-time with 500, 159, 223 fps, respectively. Table 5.2 shows the speed of all the trackers on AOLME video.

Figure 5.3 shows the performance (IoU ratios) of detector and all the fast trackers across a video. The IoU ratios are plotted for every second throughout the video. No-Data represents there is no ground truth. This figure shows that all the trackers failed with fast movements of the object. The video under consideration for the figure is 16 minutes long, having resolution of 858 × 480 @ 30 fps.


(a) Failure to detect keyboard due to lack of visible keys.



(b) False positive due to similar looking book.



(c) False positive due to similar looking book.


(d) Failure due to occlusion.

Figure 5.2: Some of the test results showing failure of keyboard detection

Table 5.2: Trackers performance using hardware system described in 5.1.2 and OpenCV 4.0. Video under consideration is 23.45 minutes long 858 × 480 @ 30 fps.

| **Method** | **FPS** | **Comments** | **Accuracy** |
|---|---|---|---|
| Boosting | 17 | Not Real Time | Not Accurate |
| MIL | 11 | Not Real Time | Not Accurate |
| Mosse | 500 | **Real time, Fastest** | **Accurate** |
| Median Flow | 223 | **Real Time** | Not Accurate |
| TLD | 21 | Not Real Time | Not Accurate |
| KCF | 159 | **Real Time** | **Accurate** |
| GOTURN | <7 | Not Real Time | Not Accurate |
| CSRT | 25 | Not Real Time | Not Accurate |

### 5.1.3 Combination of detection and tracking results

We present the results of keyboard detection (every 5 seconds) followed by tracking. Figure 5.4 shows the results of (i) detection alone and (ii) a combination of detection

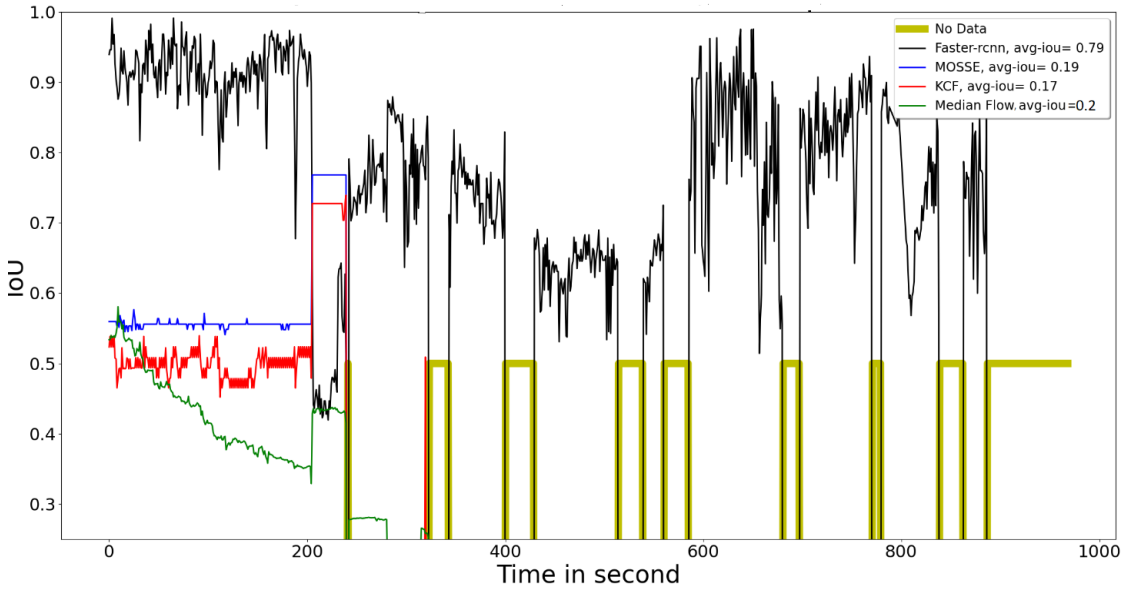



Figure 5.3: Detection vs fast-trackers.

and tracking. This method was able to achieve almost the same accuracy with a significantly faster running time. The fast tracker used to achieve this is KCF. For running detection every second on a 23-minute video, the detection alone performed at 4.7 times the real-time rate, while the combined algorithm performed at 21 times the real-time rate, maintaining the same avg-IoU ratio. Fig. 5.4 shows that detection alone is able to achieve an average IoU of 0.84 for a video and the combined system achieved an average IoU of 0.82.

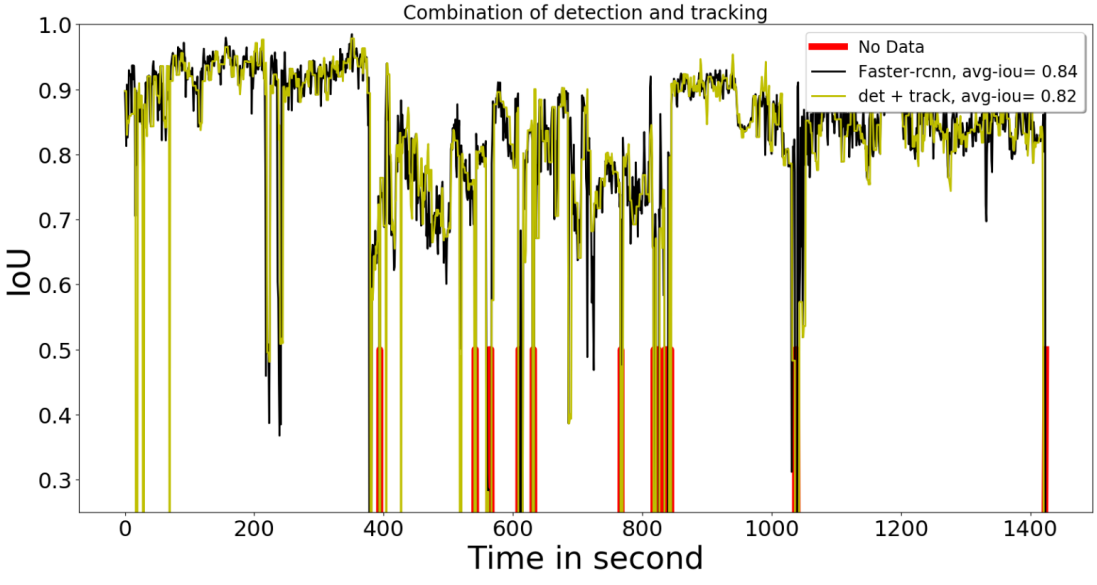



Figure 5.4: Combination of detection and tracking. Video under consideration is 23.45 minutes long having resolution of $858 \times 480$ @ 30 fps.

## 5.2 Hand detection, optimal data augmentation and projection results

This section summarizes the results of hand detection, data augmentation optimization, and projections.

### 5.2.1 Hand detection results

For hand detection, the library called MMDetection [7] is used. To run these experiments, the training time for 12 epochs is 16 minutes. The dataset used for training is shown in table 3.5. The hand detector was able to achieve Average precision (AP) of 0.72 at 0.5 intersection over union (IoU). AP @ IoU=0.50 represents the model has used 0.5 threshold value to remove unnecessary bounding boxes.

### 5.2.2 Optimal data augmentation parameters study

**Shear Angle Optimization**

All the training images are augmented with each of the below angles, and maximum validation accuracies are recorded accordingly. The angles used for experiments are 2°, 4°, 8°, 16°, and 32° . The validation accuracy started to decrease after 8°. The best validation accuracy is achieved when angle is 4°. So the range considered is (-3,3). The plot is shown in Fig. 5.5.

**Rotate Angle Optimization**

All the training images are augmented with each of the below angles and Maximum Validation accuracies are recorded accordingly. The angles used for experiments are 2°, 4°, 8°, 16°, and 32° . The validation accuracy started to decrease after 8 °. The best validation accuracy is achieved when angle is 8 °. So the range considered is

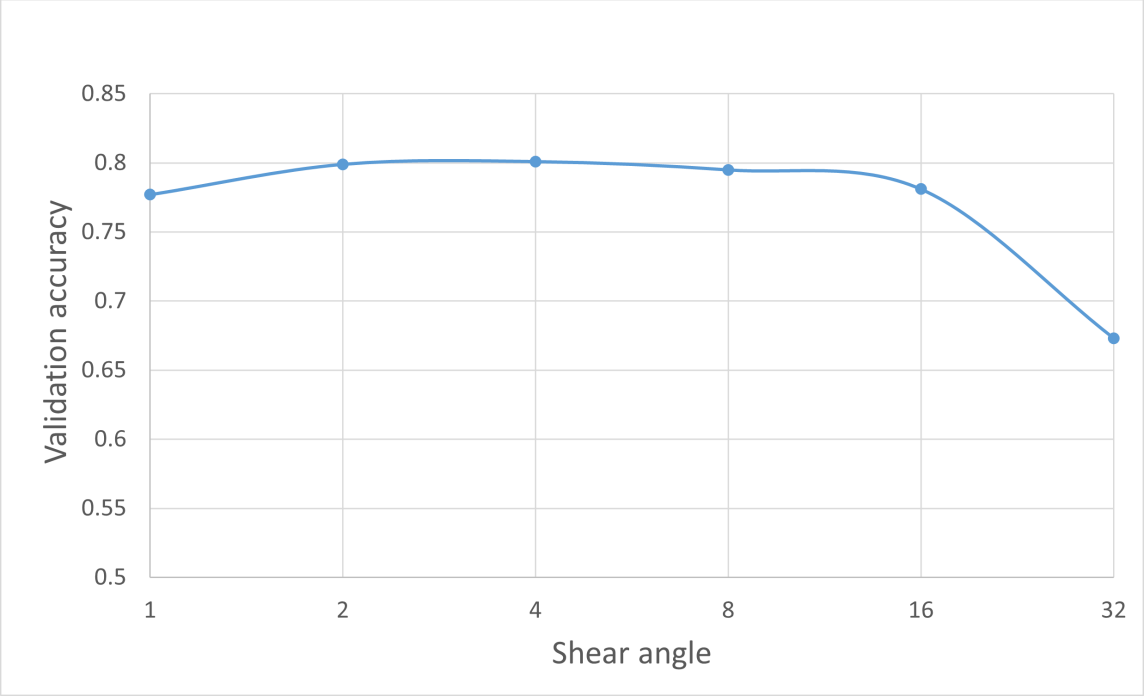



Figure 5.5: Shear angle optimization.

(-7,7). The plot is shown in figure 5.6.

**Translation Pixels Optimization**

All the training images are horizontally translated with each of the below numbers of pixels and Maximum Validation accuracies are recorded accordingly. Pixels used for Translation are 2, 4, 6, 8, 16, 32, 64, 128, 256, 512, 800 . The best validation accuracy started to decrease after 128 pixles. The best validation accuracy is when the image is translated untill 128 pixels. The plot is shown in figure 5.7. So the optimal range considered is (-20,20).

**Probability Optimization**

**Custom Augmentation** MMDetection does not support random uniform sampling. So to support random uniform sampling, custom classes for shear, rotate and translate were implemented. For each augmentation, a random uniform value is selected from the optimal ranges for shear, rotate, and translation. The value is made

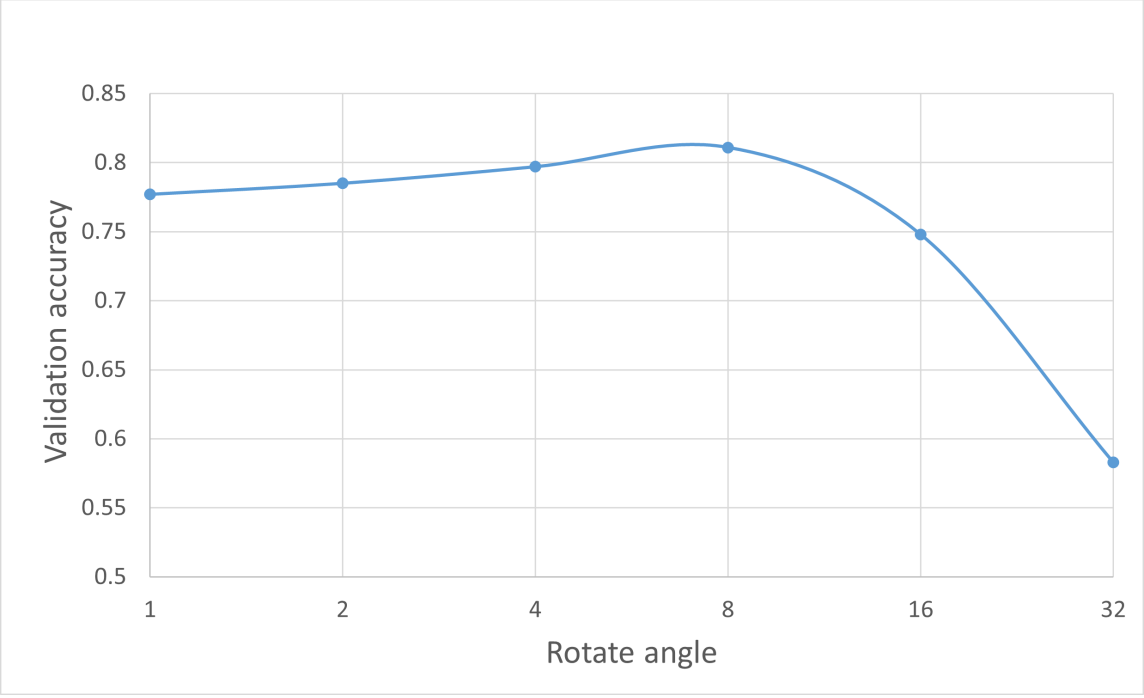



Figure 5.6: Rotate angle optimization.

negative with half the probability.

To find the optimal probability, five different augmentations are used. The order of these augmentations are listed in table 5.3.

Table 5.3: Augmentations used for probability study

| Data augmentation method | Parameter range |
|---|---|
| H-flip | |
| Rescale | [0.8,1.2] |
| Shear | [-3,3] |
| Rotate | [-7,7] |
| Translate | [-20,20] |

The probabilities considered for the experiments are P ={0, 0.25, 0.5, 0.75, 1.0}. The figure for all the probabilities considered is shown in Fig. 5.8. The best validation accuracy is with probability 0.5.

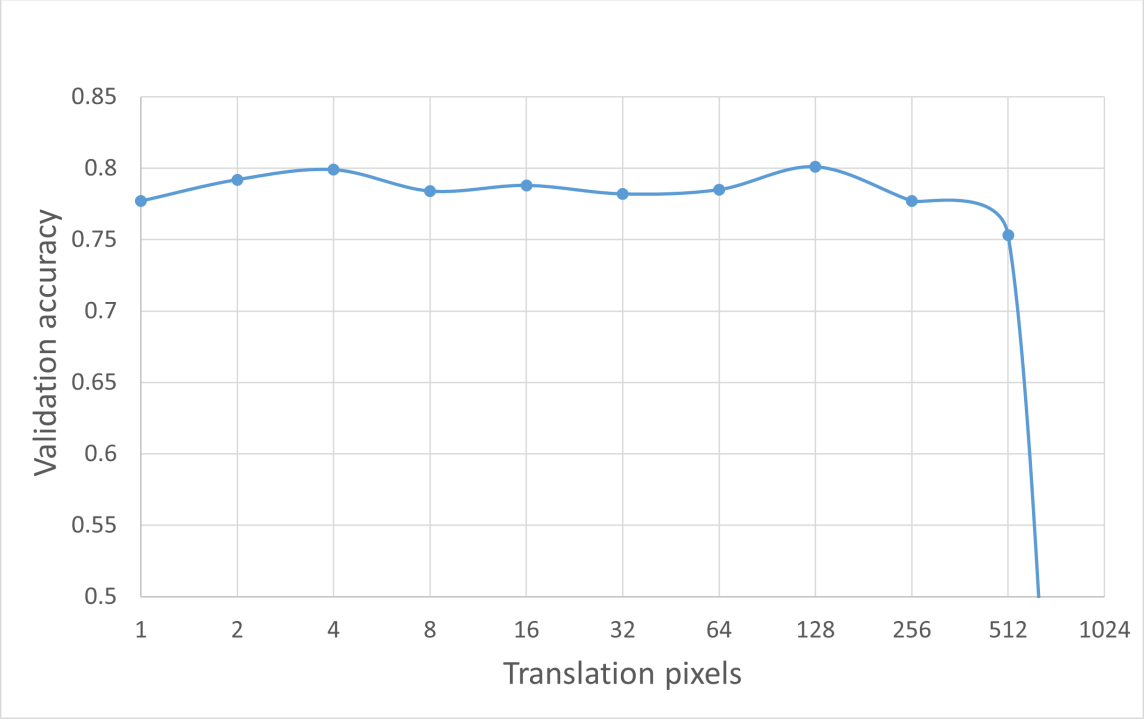



Figure 5.7: Translation pixels optimization.

Now that all the optimal parameters are obtained, model is trained with the optimized dataset, the improvement of validation accuracies of no-augmentation vs all the five augmentation applied with 0.5 probability is shown in Fig. 5.9

### 5.2.3 Improvement of Hand Detection Results using Optimal Data Augmentation

An AP of 0.81 was achieved by training the model with optimized data augmentation parameters. The training time remains the same as the size of dataset is not changed. Table 5.4 summarizes the final results of no augmentation vs optimal augmentation. An improvement of AP of 11% and 9% on the validation and testing set, respectively, was achieved. Here the word best means the epoch, which gave maximum validation accuracy.

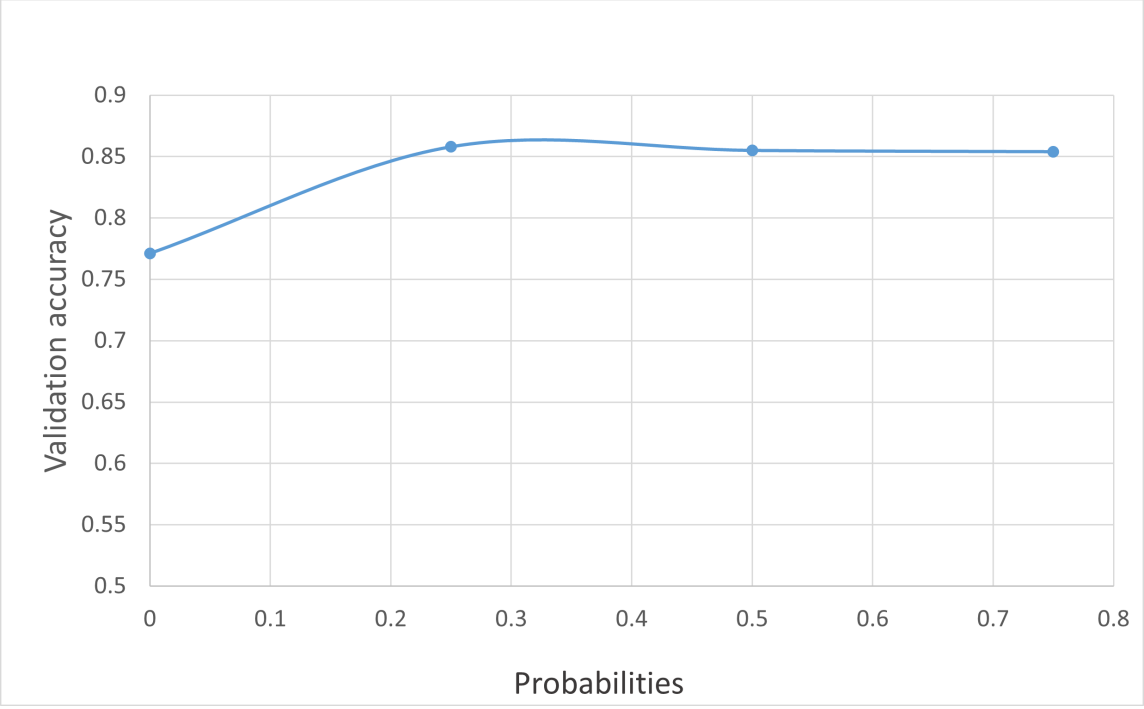



Figure 5.8: Validation Accuracy with Probability = 0.5.

Table 5.4: Validation and Testing Accuracies for multiple probabilities.

| **Data Split** | **Model** | **No-aug** | **P=0.25** | **P =0.5** | **P=0.75** | **P=1.0** |
|---|---|---|---|---|---|---|
| **Val** | Best | 0.77 | 0.86 | **0.86** | 0.85 | 0.84 |
| | Last | 0.76 | 0.85 | **0.86** | 0.84 | 0.82 |
| **Test** | Best | 0.75 | 0.80 | **0.80** | 0.79 | 0.78 |
| | Last | 0.71 | 0.80 | **0.81** | 0.78 | 0.76 |

Fig. 5.10 shows some of the testing results for hand detection.

As seen in Fig. 5.10, it is evident that there are many hands detected in the picture. The main goal here is to propose activity regions for writing within the primary table of focus; I need to eliminate the background hands. The other goal is to reduce the number of regions proposed to classify.

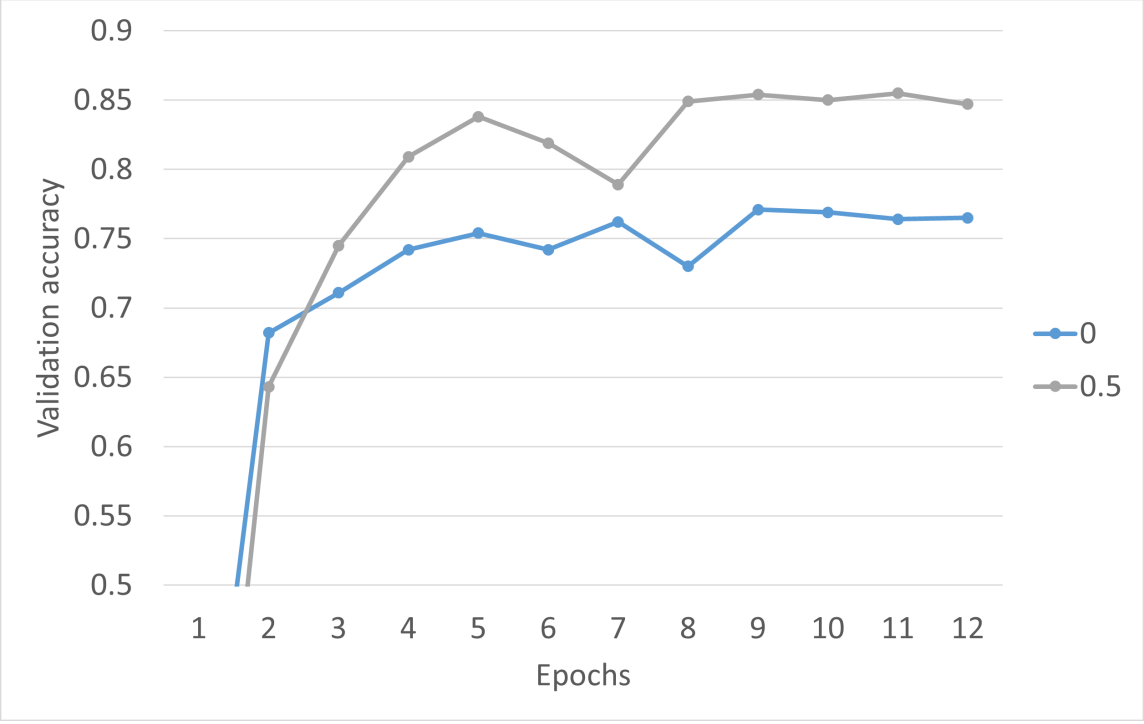



Figure 5.9: Probability Study.

### 5.2.4 Performance Evaluation Protocol of Using Projections

Fig. 5.11 shows how the performance evaluation is performed. The boxes marked in blue are results of hand detection and the green box represents ground truth for writing instances.

Steps involved in performance evaluation are:

- Classify each second as writing instance if more than half of the frames are labeled as writing in ground Truth.
- Take all the hand proposals.
- Calculate IoU ratio for each proposal region w.r.t writing ground truth.
- Take the value which corresponds to maximum IoU ratio.

(a) (b) (c) (d) (e) (f)

Figure 5.10: Examples of hand detection.

Once all the regions and corresponding IoU ratios are obtained, they are plotted using box plot as shown in figure 5.13.

**Naive Region Proposals:**

Naive region proposal approach is considered to take all the hand detections generated by Faster R-CNN into consideration.

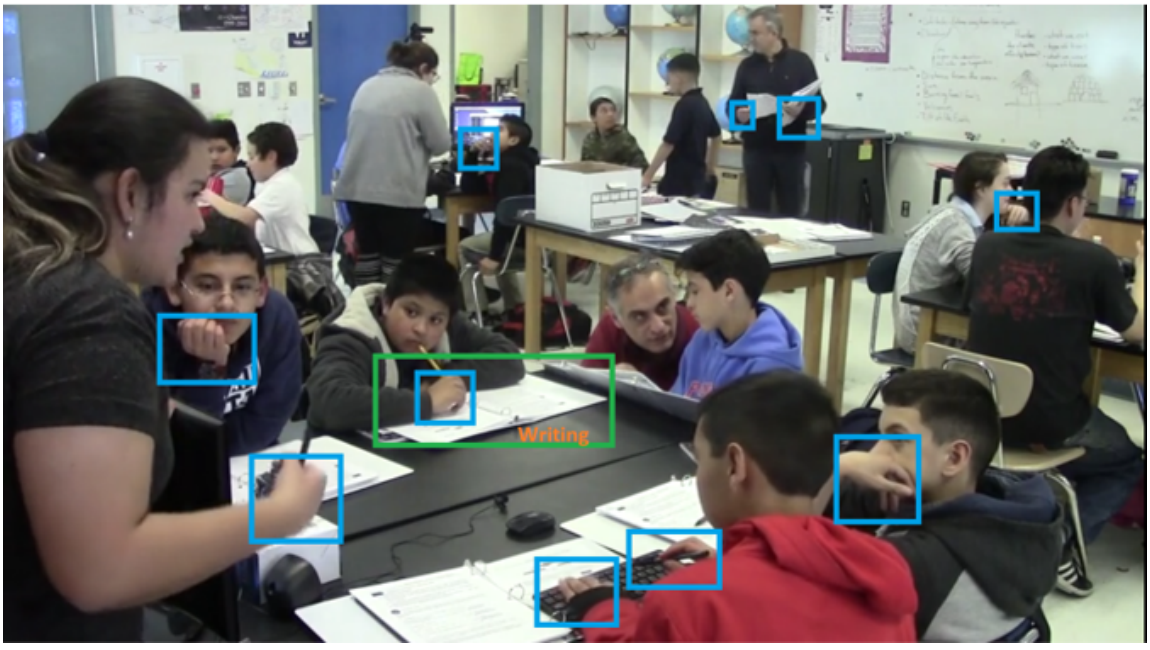



Figure 5.11: Calculation of performance evaluation.

Figure 5.12 shows the results on video frames. The figures on the left use the Naive approach, and the ones on the right use Projections. These test frames clearly show the reduction in the number of regions and capturing all writing instances.

Figure 5.13 shows the performance of two approaches i) Naive and ii) Projections for all the test sessions. IoU ratios are calculated for every writing instance with the proposals generated by the Naive method and Projections. The approach was very effective in removing false positive detections that correspond to hands from a different group. Yet, the approach was able to detect all the hands from the collaborative group that was closer to the camera (as required).

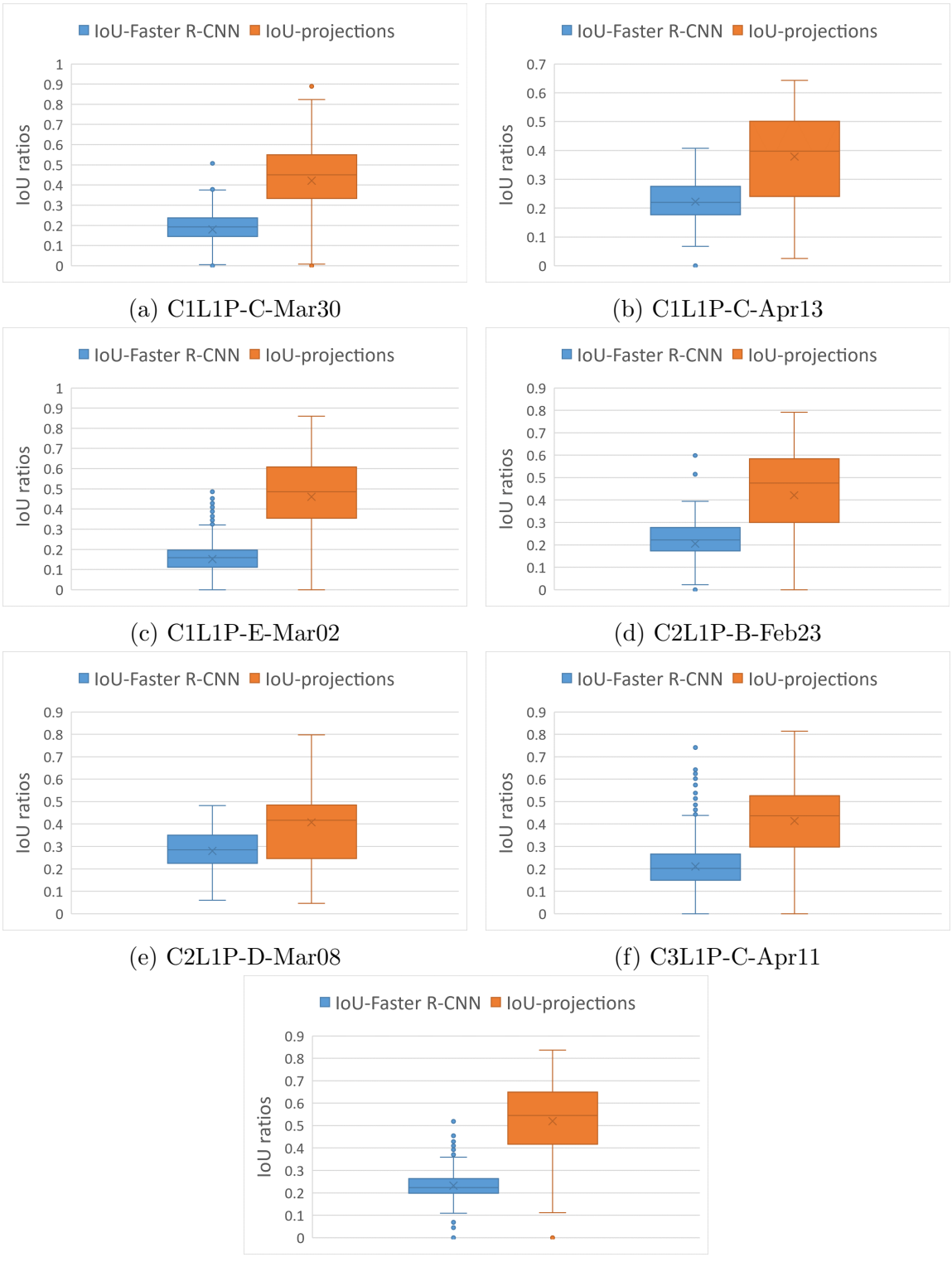
Table 5.5 shows the percentage of reduction in the number of proposal regions for the two approaches for all the test sessions. The projection approach was able to achieve an average of 80% reduction in the proposed activity regions and also capture all the instances of collaborative group shown in Fig. 5.13.

(a) (b)

(c) (d)

(e) (f)

Figure 5.12: Results showing the reduction of proposal regions and capturing writing instances. The ones on the left are original and the ones on right are after projections and segmentation.




(a) C1L1P-C-Mar30

(b) C1L1P-C-Apr13

(c) C1L1P-E-Mar02

(d) C2L1P-B-Feb23

(e) C2L1P-D-Mar08

(f) C3L1P-C-Apr11

(g) C3L1P-D-Feb21

Figure 5.13: Performance evaluation for all the test sessions.

Table 5.5: Reduction in number of region proposals for each test session.

| Group | Date | Naive | Using Projections | % Reduction |
|---|---|---|---|---|
| C1L1P-C | Mar30 | 55914 | 9804 | **82.5** |
| C1L1P-C | Apr13 | 34665 | 8028 | **76.8** |
| C1L1P-E | Mar02 | 50312 | 9968 | **80.0** |
| C2L1P-B | Feb23 | 48073 | 9924 | **79.3** |
| C2L1P-D | Mar08 | 31875 | 7724 | **75.7** |
| C3L1P-C | Apr11 | 36757 | 9536 | **74.0** |
| C3L1P-D | Mar19 | 57319 | 9536 | **83.3** |

# Chapter 6

# Conclusion and Future Work

## 6.1 Conclusion

In this thesis, a new method was proposed which uses projections and cluster-based segmentation to identify potential activity regions, which include hand movements. The primary contributions of this thesis include: (i) Robust detection and fast-tracking of the keyboard, (ii) a detailed optimal data augmentation parameter study showing an improvement over detection results, and (iii) a novel method that uses projections and cluster-based segmentation to identify hand regions of the current collaborative group. In each case, the combined algorithm of detection and fast-tracking achieved almost the same accuracy with a lot more speed up of time. The model trained with optimal data augmentation parameters achieved an improvement of 8% over detection results. Furthermore, the novel method was able to capture all the writing activity regions by reducing the number of region proposals by 80%.

## 6.2 Future Work

Using the proposed methods, we found improvements in:

- **Region Proposals for Activity Recognition**. This thesis, combined with activity detection algorithms developed in ivpcl, would allow a computer-aided analysis tool to find activities of interest for education researchers, such as typing or writing. The block diagram to generate group activity maps is shown in Fig. 6.1. To generate activity proposals for typing detection, combined method of keyboard detection and tracking is used. Similarly, to generate activity proposals for writing detection, combination of hand detection and projections is used. The student activity map generated using the approach below is shown in Fig. 6.2. The activity map shows each student's participation with respect to typing for a session of 90 minutes long.

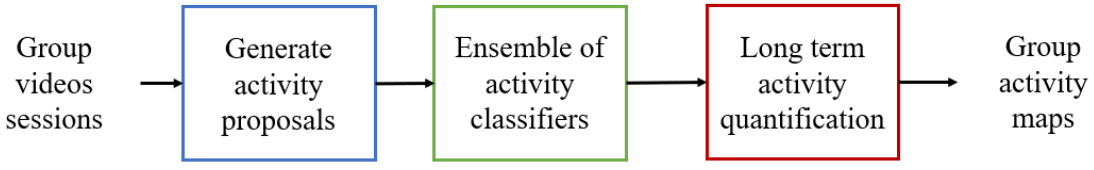



Figure 6.1: Block diagram to generate activity maps

- **Matching hand regions to students**. The hand activity clusters can be matched to students in a session to generate student participation maps as shown in 6.3 and also to associate activities for each student. The map shows participation of each student in a session which is 90 minutes long.

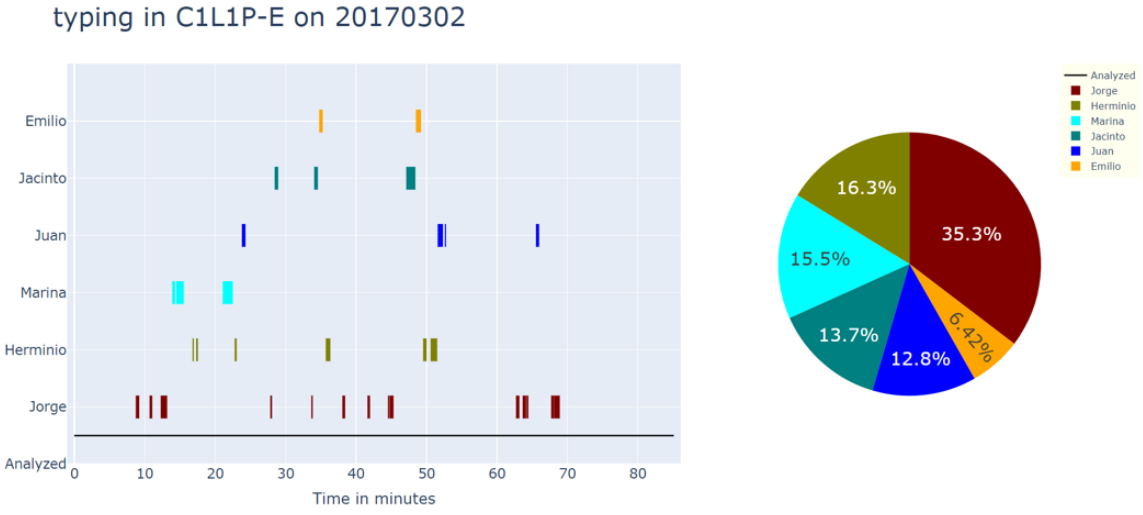
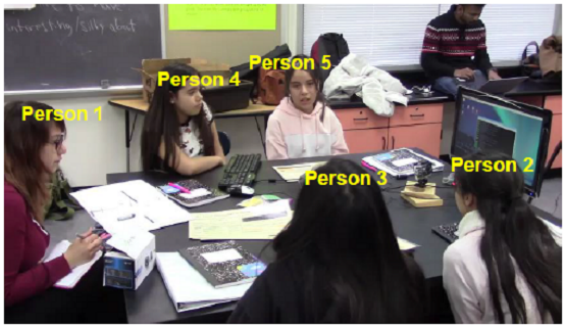
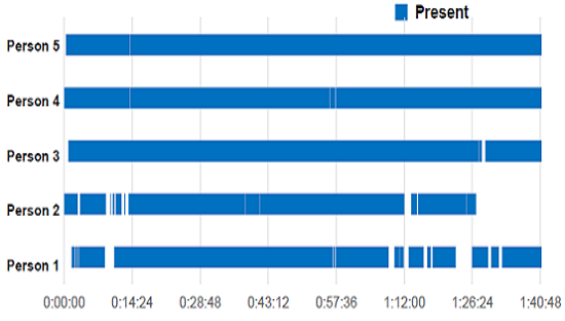



Figure 6.2: Session typing map




(a) Frame extracted from student interactions.




(b) Student participation map.

Figure 6.3: Participation map for a session.

# References

[1] Lubna Aziz, Sah bin Haji Salam, and Sara Ayub, *Exploring deep learning-based architecture, strategies, applications and current trends in generic object detection: A comprehensive review*, IEEE Access (2020).

[2] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie, *Visual tracking with online multiple instance learning*, 2009 IEEE Conference on computer vision and Pattern Recognition, IEEE, 2009, pp. 983–990.

[3] Geoffrey H Ball and David J Hall, *Isodata, a novel method of data analysis and pattern classification*, Tech. report, Stanford research inst Menlo Park CA, 1965.

[4] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui, *Visual object tracking using adaptive correlation filters*, 2010 IEEE computer society conference on computer vision and pattern recognition, IEEE, 2010, pp. 2544–2550.

[5] Cesar Carranza, Daniel Llamocca, and Marios Pattichis, *Fast and scalable 2d convolutions and cross-correlations for processing image databases and videos on cpus*, 2020 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI), IEEE, 2020, pp. 70–73.

[6] Alvaro E Ulloa Cerna, Linyuan Jing, Christopher W Good, Sushravya Raghunath, Jonathan D Suever, Christopher D Nevius, Gregory J Wehner, Dustin N Hartzel, Joseph B Leader, Amro Alsaid, et al., *Deep-learning-assisted analysis of echocardiographic videos improves predictions of all-cause mortality*, Nature Biomedical Engineering (2021), 1–9.

[7] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al., *Mmdetection: Open mmlab detection toolbox and benchmark*, arXiv preprint arXiv:1906.07155 (2019).

[8] Callie J Darsey, *Hand movement detection in collaborative learning environment videos*, (2018).

[9] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé, *Mot20: A benchmark for multi object tracking in crowded scenes*, arXiv preprint arXiv:2003.09003 (2020).

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, *Imagenet: A large-scale hierarchical image database*, 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.

[11] Cody W Eilar, Venkatesh Jatla, Marios S Pattichis, Carlos LópezLeiva, and Sylvia Celedón-Pattichis, *Distributed video analysis for the advancing out of school learning in mathematics and engineering project*, 2016 50th Asilomar Conference on Signals, Systems and Computers, IEEE, 2016, pp. 571–575.

[12] Cody Wilson Eilar, *Distributed and scalable video analysis architecture for human activity recognition using cloud services*, (2016).

[13] Gangadharan Esakki, *Adaptive encoding for constrained video delivery in hevc, vp9, av1 and vvc compression standards and adaptation to video content*, Ph.D. thesis, The University of New Mexico, 2020.

[14] Gangadharan Esakki, Venkatesh Jatla, and Marios S Pattichis, *Optimal hevc encoding based on gop configurations*, 2016 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI), IEEE, 2016, pp. 25–28.

[15] Gangadharan Esakki, Venkatesh Jatla, and Marios S. Pattichis, *Adaptive high efficiency video coding based on camera activity classification.*, DCC, 2017, p. 438.

[16] Gangadharan Esakki, Andreas Panayides, Sravani Teeparthi, and Marios Pattichis, *A comparative performance evaluation of vp9, x265, svt-av1, vvc codecs leveraging the vmaf perceptual quality metric*, Applications of Digital Image Processing XLIII, vol. 11510, International Society for Optics and Photonics, 2020, p. 1151010.

[17] Gangadharan Esakki, Andreas S Panayides, Venkatesh Jalta, and Marios S Pattichis, *Adaptive video encoding for different video codecs*, IEEE Access **9** (2021), 68720–68736.

[18] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman, *The pascal visual object classes challenge 2007 (voc2007) results*, (2007).

[19] Mark Everingham and John Winn, *The pascal visual object classes challenge 2012 (voc2012) development kit*, Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep **8** (2011).

[20] Helmut Grabner, Michael Grabner, and Horst Bischof, *Real-time tracking via on-line boosting.*, Bmvc, vol. 1, Citeseer, 2006, p. 6.

[21] David Held, Sebastian Thrun, and Silvio Savarese, *Learning to track at 100 fps with deep regression networks*, European conference on computer vision, Springer, 2016, pp. 749–765.

[22] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista, *High-speed tracking with kernelized correlation filters*, IEEE transactions on pattern analysis and machine intelligence **37** (2014), no. 3, 583–596.

[23] Abigail R Jacoby, *Context-sensitive human activity classification in video utilizing object recognition and motion estimation*, (2017).

[24] Abigail Ruth Jacoby, Marios S Pattichis, Sylvia Celedón-Pattichis, and Carlos LópezLeiva, *Context-sensitive human activity classification in collaborative learning environments*, 2018 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI), IEEE, 2018, pp. 1–4.

[25] Venkatesh Jatla, *Automatic segmentation of coronal holes in solar images and solar prediction map classification*, (2016).

[26] Venkatesh Jatla, Marios S Pattichis, and Charles Nick Arge, *Image processing methods for coronal hole segmentation, matching, and map classification*, IEEE Transactions on Image Processing **29** (2019), 1641–1653.

[27] Venkatesh Jatla, Sravani Teeparthi, Marios S. Pattichis, Sylvia Celedón-Pattichis, and Carlos López Leiva, *Long-term human video activity quantification of student participation*, 2021 55th Asilomar Conference on Signals, Systems, and Computers, IEEE, 2021.

[28] Zdenek Kalal, Krystian Mikolajczyk, and Matas Jiri, *Tracking-learning-detection*, IEEE transactions on pattern analysis and machine intelligence **34** (2011), no. 7, 1409–1422.

[29] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas, *Forward-backward error: Automatic detection of tracking failures*, 2010 20th international conference on pattern recognition, IEEE, 2010, pp. 2756–2759.

[30] Robert B Kent and Marios S Pattichis, *Design, implementation, and analysis of high-speed single-stage n-sorters and n-filters*, IEEE Access (2020).

[31] Matej Kristan, Ales Leonardis, Jiri Matas, Michael Felsberg, Roman Pflugfelder, Luka ˇCehovin Zajc, Tomas Vojir, Goutam Bhat, Alan Lukezic, Abdelrahman Eldesokey, et al., *The sixth visual object tracking vot2018 challenge results*, Proceedings of the European Conference on Computer Vision (ECCV) Workshops, 2018, pp. 0–0.

[32] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, *Microsoft coco: Common objects in context*, European conference on computer vision, Springer, 2014, pp. 740–755.

[33] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg, *Ssd: Single shot multibox detector*, European conference on computer vision, Springer, 2016, pp. 21–37.

[34] Alan Lukezic, Tomas Vojir, Luka ˇCehovin Zajc, Jiri Matas, and Matej Kristan, *Discriminative correlation filter with channel and spatial reliability*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 6309–6318.

[35] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler, *Mot16: A benchmark for multi-object tracking*, arXiv preprint arXiv:1603.00831 (2016).

[36] Matthias Mueller, Neil Smith, and Bernard Ghanem, *A benchmark and simulator for uav tracking*, European conference on computer vision, Springer, 2016, pp. 445–461.

[37] Matthias Muller, Adel Bibi, Silvio Giancola, Salman Alsubaihi, and Bernard Ghanem, *Trackingnet: A large-scale dataset and benchmark for object tracking in the wild*, Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 300–317.

[38] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, *You only look once: Unified, real-time object detection*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.

[39] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, *Faster r-cnn: towards real-time object detection with region proposal networks*, IEEE transactions on pattern analysis and machine intelligence **39** (2016), no. 6, 1137–1149.

[40] WENJING SHI, *Human attention detection using am-fm representations*, (2016).

[41] Wenjing Shi, Marios S. Pattichis, Sylvia Celedón-Pattichis, and Carlos López Leiva, *Person detection in collaborative group learning environments using multiple representations*, 2021 55th Asilomar Conference on Signals, Systems, and Computers, IEEE, 2021.

[42] Wenjing Shi, Marios S Pattichis, Sylvia Celedón-Pattichis, and Carloz López Leiva, *Talking detection in collaborative learning environments*, 19th International Conference CAIP, Springer, 2021.

[43] Wenjing Shi, Marios S Pattichis, Sylvia Celedón-Pattichis, and Carlos LópezLeiva, *Robust head detection in collaborative learning environments using am-fm representations*, 2018 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI), IEEE, 2018, pp. 1–4.

[44] Piotr Skalski, *Make Sense*, `https://github.com/SkalskiP/make-sense/`, 2019.

[45] Luis Sanchez Tapia, Marios S. Pattichis, Sylvia Celedón-Pattichis, and Carlos López Leiva, *Bilingual speech recognition by estimating speaker geometry from video data*, 19th International Conference CAIP, Springer, 2021.

[46] Luis Sanchez Tapia, Marios S Pattichis, Sylvia Celedón-Pattichis, and Carlos López Leiva, *The importance of the instantaneous phase for face detection using simple convolutional neural networks*, 2020 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI), IEEE, 2020, pp. 1–4.

[47] Sravani Teeparthi, Venkatesh Jatla, Marios S. Pattichis, Sylvia Celedón-Pattichis, and Carlos López Leiva, *Fast hand detection in collaborative learning environments*, 19th International Conference CAIP, Springer, 2021.

[48] Phuong Tran, Marios S. Pattichis, Sylvia Celedón-Pattichis, and Carlos López Leiva, *Facial recognition in collaborative learning videos*, 19th International Conference CAIP, Springer, 2021.

[49] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu, *scikit-image: image processing in python*, PeerJ **2** (2014), e453.

[50] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr, *Fast online object tracking and segmentation: A unifying approach*, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 1328–1338.

[51] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick, *Detectron2 (2019)*, URL https://github.com/facebookresearch/detectron2.

[52] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu, *Fairmot: On the fairness of detection and re-identification in multiple object tracking*, arXiv preprint arXiv:2004.01888 (2020).