ECSE 415 Final Project: Detecting, Localizing, and Tracking People

Kaiyue Pan
McGill University
Montreal, Canada
kaiyue.pan@mail.mcgill.ca

Ruoli Wang
McGill University
Montreal, Canada
ruoli.wang@mail.mcgill.ca

Weixin Wu
McGill University
Montreal, Canada
weixin.wu@mail.mcgill.ca

Xuemeng Sun
McGill University
Montreal, Canada
xuemeng.sun@mail.mcgill.ca

Yuhang Zhang
McGill University
Montreal, Canada
yuhang.zhang@mail.mcgill.ca

Abstract

Given a subset of the MOT as training and testing dataset, the final project requires students to implement a complete pipeline that detects, localizes and tracks individuals walking across a scene from a fixed viewpoint. The pipeline includes dataset setup, a classifier, a localizer and an object tracker. The tools used in this project include Jupyter Notebook with python3 programming language.

1. Introduction

Object identification has been a long-standing challenge in the field of machine learning and computer vision. With the advancement in hardware and the rise of deep learning and other image processing techniques, computers have gradually mastered the tasks of object classification, detection, and localization. The growth in computer vision, along with the convergence of advanced imaging analysis, robotics, and related automation, is now leading to creative, never-before-seen solutions in different industries.

In this final project, we have the opportunity to apply the different concepts/techniques we learned in this class and build a complete pipeline that detects, localizes, and tracks individuals walking across a scene from a fixed viewpoint.

An overview of our pipeline is as follows: using different parts of the Multiple Object Tracking Benchmark (MOT) as training and validation set, we begin by extracting features from different frames using HoG (Histogram of Oriented Gradients) and training an SVM (Support Vector Machines) classifier that can discriminate between two classes, a person class and a non-person class. Then, the classifier is used to localize all instances of people in different frames. Finally, instances of people are tracked from frame to frame in a video sequence using the Optical flow and the MOSSE tracker. The implementation and results for each stage will be explained in the upcoming sections of the report.

2. Dataset

The training dataset contains two video sequences from MOT15, namely TUD-Campus and TUD-Stadtmitte. All frames are given at fixed resolution of 640x480 with fixed perspectives. By preprocessing the 71 frames from TUD-Campus and the 179 frames from TUD-Stadtmitte, we are able to extract 1515 person data. The person data is extracted from each frame based on the bounding box annotations provided in the ground truth (gt.txt). For consistency and efficiency, every person's data is then resized to the dimensions of **56x174** pixels, which is the average dimensions for all person data. Figure 2.1 shows an example of the person data extracted from a frame in the Campus dataset.

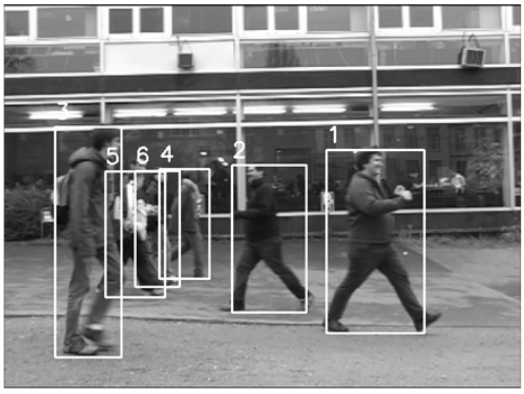


Figure 2.1: Person data extracted from a frame in the Campus dataset

Moreover, the project description specifically mentions that we need to make sure that "the same person, regardless of frame, does not exist in both training and validation sets" for k-fold cross-validation. To do so, we preprocess the person data obtained by partitioning the person data list into k sublists, where k is the number of different subjects found in the training dataset. The value of k can be determined from the value of the "id" field given in the ground truth. By using each of the k sublists as either the training set or the validation set, we ensure that there is no overlap between the two sets during k-fold cross-validation. The details regarding our k-fold cross-validation will be provided in Section 3.2 of this report.

On the other hand, we are able to extract 67 non-person data from the given dataset. The non-person data is selected to ensure minimal intersection with the people bounding boxes. The non-person data is also resized to **56x174** pixels for consistency and efficiency.

Figure 2.2: Non-person data extracted from a frame in the Campus dataset

Something worth pointing out is that we initially did not include a variety of non-person data. As a result, the trained classifier did not produce good results during the detection and localization phase. As shown in Figure 2.3, label 3 and label 6 are classified as "person"because there are people sitting behind the windows.

Figure 2.3: Detection and Localization results before we updated our non-person dataset

However, after adding the "glass window" patches (such as labels 1, 2, 3 in Figure 2.2) into the ground truth for non-person, the classifier's performance improved after training. Please refer to the detection and localization results shown in Figure 2.4 for more details.

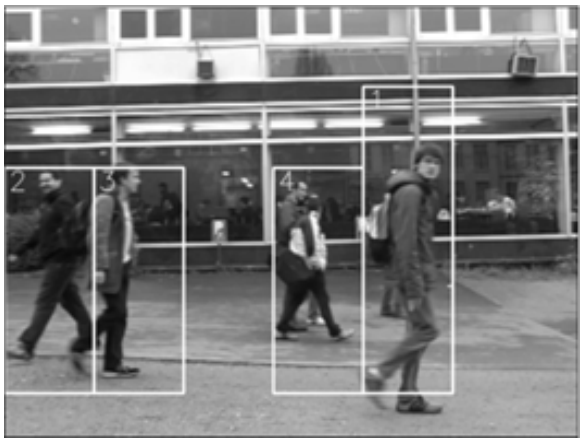


Figure 2.4: Detection and Localization results after we updated our non-person dataset

3. Classification

3.1 Feature Extraction

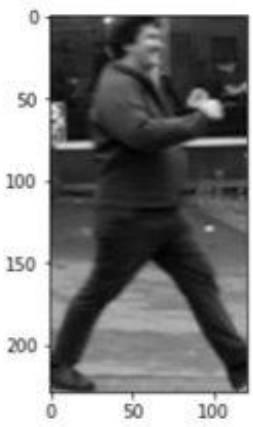


Figure 3.1.1: Original person data/image

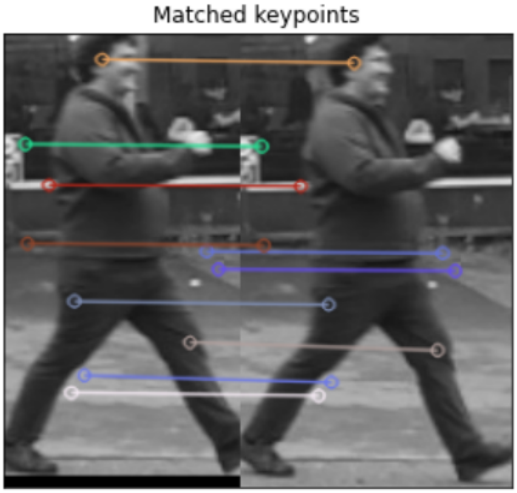


Figure 3.1.2: An example of SIFT features on the person data/image

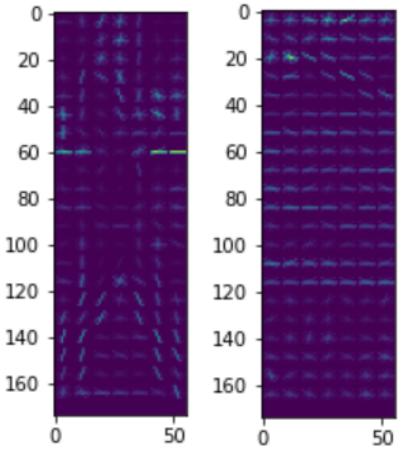


Figure 3.1.3: HoG features of a person (left) and a non-person (right)

For feature extraction, our first approach is to use SIFT features, as shown in Figure 3.1.2. However, SIFT extracted feature pixels that are in the background, which would fail to distinguish non-person images from person images regardless of the number of features. Moreover, SIFT features require matching and sorting before we choose thresholded features. Hence, we implement HoG, which leads to reasonable test results. In our tests, HoG features could capture the contour of each person, as shown in Figure 3.1.3.

To ensure that we could get the same number of features for all input images, we resized each image to be of the mean size of the dataset (56x174 pixels).

As shown in Figure 3.1.4, there are four parameters for the HoG function: the number of orientations, cell size, block size, and a scaling factor to scale the images during the feature extraction process. We focus on testing 2 of the parameters, namely the number of orientations and the scale factor of images. These two parameters would affect the number of features extracted from each image.

```
def HOG(input_list,N_orientations,pixels_per_cell,cells_per_block,scaling_factor)
  output_h = []
  output_img = []
  for i in range(len(input_list)):
    resized_img = cv2.resize(input_list[i],None,fx = scaling_factor,
                         fy = scaling_factor, interpolation = cv2.INTER_LINEAR)
    fd, hog_image = hog(resized_img, orientations=N_orientations,
                        pixels_per_cell = pixels_per_cell,
          cells_per_block=cells_per_block, visualize=True, multichannel=False)
    output_h.append(fd)
    output_img.append(hog_image)
  return output_h,output_img
```

Figure 3.1.4: HoG function

3.2 K-fold Cross-validation

Cross-validation is a resampling procedure for evaluating machine learning models on a limited data sample. The procedure has a single parameter called k, which refers to the number of groups/bins that a given data sample is split into. Cross-validation would estimate the performance of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how well the model is performing, in general, when it is used to make predictions on data the classifier has not "seen" during the training of the model.

The general procedure is as follows:

1. Shuffle the dataset randomly. Therefore, we get different splits for each trial of the tests.

2. Split the dataset into k groups. In our case, since our dataset is 2D, and each sublist corresponds to the person data/image for one particular person, we are arranging the sublists into K groups to ensure the training set and test set contain different people.
3. For each unique group:
   a. Take the group as a holdout or test data set.
   b. Take the remaining groups as a training data set.
   c. Fit a model on the training set and evaluate the model's performance on the test set.
   d. Retain the evaluation score and discard the model.
4. Summarize the skill of the model using the sample of model evaluation scores.

3.3 Classifier Validation Tests

For validation, we test on 2 classifiers: the linear SVM classifier and the K-nearest neighbor classifier. We optimize four hyperparameters in total: number of orientations and image size at feature extraction stage, C (regularization parameter) for the SVM classifier, and K (number of nearest neighbors) for the K-nearest neighbor classifier. The input image size in HoG is *the scaling factor*average size of the original dataset*.

For each test, we first run the two classifiers with the same hyperparameters of the HoG function. We select the appropriate values for hyperparameters based on three metrics: accuracy, precision, and recall. The word "average" in the test figures indicates that it is the average number from the 5-fold cross-validation.

Test 1: 9 orientations, scaling factor = 1.

```
Average Accuracy:  98.82   Standard deviation: 1.04
Average Precision: 99.54   Standard deviation: 0.75
Average Recall:    99.17   Standard deviation: 1.23
```

Figure 3.3.1: SVM, C = 1

```
Average Accuracy:  99.66   Standard deviation: 0.69
Average Precision: 99.62   Standard deviation: 0.77
Average Recall:    100.0   Standard deviation: 0.0
```

Figure 3.3.2: KNN, K = 1

Test 2: 12 orientations, scaling factor = 1.

```
Average Accuracy:  98.99   Standard deviation: 0.72
Average Precision: 99.54   Standard deviation: 0.75
Average Recall:    99.36   Standard deviation: 0.8
```

Figure 3.3.3: SVM, C = 1

```
Average Accuracy:  99.66   Standard deviation: 0.69
Average Precision: 99.62   Standard deviation: 0.77
Average Recall:    100.0   Standard deviation: 0.0
```

Figure 3.3.4: KNN, K = 1

As shown in Figure 3.3.1 and Figure 3.3.3, when we increase the number of orientations, SVM classifier performs slightly better than KNN. However, the test would take longer to finish, since there are more features to train.

Test 3: 9 orientations, scaling factor = 0.6.

```
Average Accuracy:  98.99   Standard deviation: 0.72
Average Precision: 99.54   Standard deviation: 0.75
Average Recall:    99.36   Standard deviation: 0.8
```

Figure 3.3.5: SVM, C = 1

```
Average Accuracy:  99.66   Standard deviation: 0.69
Average Precision: 99.62   Standard deviation: 0.77
Average Recall:    100.0   Standard deviation: 0.0
```

Figure 3.3.6: KNN, K = 1

Test 4: 9 orientations, scaling factor = 0.3.

```
Average Accuracy:  99.44   Standard deviation: 0.42
Average Precision: 99.57   Standard deviation: 0.38
Average Recall:    99.82   Standard deviation: 0.25
```

Figure 3.3.7: SVM, C = 1

```
Average Accuracy:  99.83   Standard deviation: 0.34
Average Precision: 99.81   Standard deviation: 0.39
Average Recall:    100.0   Standard deviation: 0.0
```

Figure 3.3.8: KNN, K = 1

As shown in Figure 3.3.1, Figure 3.3.5 and Figure 3.3.7, when we reduce the image size, the performance of both classifiers, overall, becomes better. In addition, the computational time can be greatly reduced as well.

The following tests focus on parameters of the classifiers. For KNN, K cannot exceed 13. This is because we have 67 non-person images, and each validation set contains 13 of them on average, and we expect the K nearest neighbors of one particular image to be either person or non-person.

Test 5: 9 orientations, scaling factor = 1.

```
Average Accuracy:  98.82   Standard deviation: 1.04
Average Precision: 99.54   Standard deviation: 0.75
Average Recall:    99.17   Standard deviation: 1.23
```

Figure 3.3.9: SVM, C = 10 (increased from 1)

```
Average Accuracy:  99.66   Standard deviation: 0.69
Average Precision: 99.62   Standard deviation: 0.77
Average Recall:    100.0   Standard deviation: 0.0
```

Figure 3.3.10: KNN, K = 3 (increased from 1)

Test 6: 9 orientations, scaling factor = 1.

```
Average Accuracy:  98.94   Standard deviation: 0.86
Average Precision: 99.54   Standard deviation: 0.75
Average Recall:    99.3    Standard deviation: 0.99
```

Figure 3.3.11: SVM, C = 0.1 (decreased from 1)

```
Average Accuracy:  99.66   Standard deviation: 0.69
Average Precision: 99.62   Standard deviation: 0.77
Average Recall:    100.0   Standard deviation: 0.0
```

Figure 3.3.12: KNN, K = 10 (increased from 1)

As shown in Figure 3.3.1, Figure 3.3.9 and Figure 3.3.11, the result from C = 0.1 is slightly better than C = 1 and C= 10 for the SVM classifier. For the KNN classifier, the result remains the same for K = 1, 3 and 10.

It can be easily seen that the overall performance of the K-nearest neighbor classifier is slightly better than the SVM classifier for the above cases. Since the dataset is randomly splitted for each trial, the test results would have subtle changes. However, due to the fact that we have limited non-person images for training and the overall performance for the SVM classifier is reasonable, we choose to proceed with the SVM classifier for the detection and localization phase.

3.4 Classifier Evaluation

```
2 kfold_crossvalid(label_train, data_train)
```

```
%Test Accuracy: 100.0
%Test Precision: 100.0
%Test Recall: 100.0
%Test Accuracy: 99.02912621359224
%Test Precision: 99.33993399339934
%Test Recall: 99.66887417218543
%Test Accuracy: 100.0
%Test Precision: 100.0
%Test Recall: 100.0
%Test Accuracy: 100.0
%Test Precision: 100.0
%Test Recall: 100.0
%Test Accuracy: 99.35275080906149
%Test Precision: 99.33774834437087
%Test Recall: 100.0
```

Figure 3.4: Test result of k-fold cross-validation

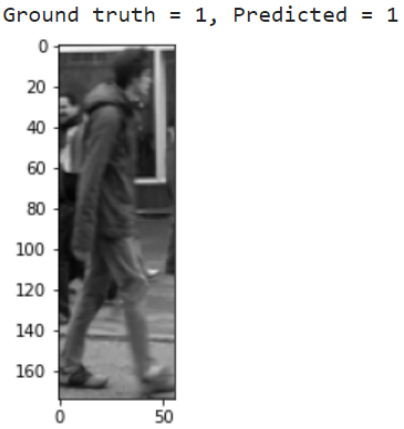
As shown in Figure 3.4, the result for the k-fold cross-validation suggests that the classifier performs well on the dataset in terms of the accuracy, precision, and the recall value. Therefore, we understand that our tuned SVM classifier is a reasonable modeler of the data.

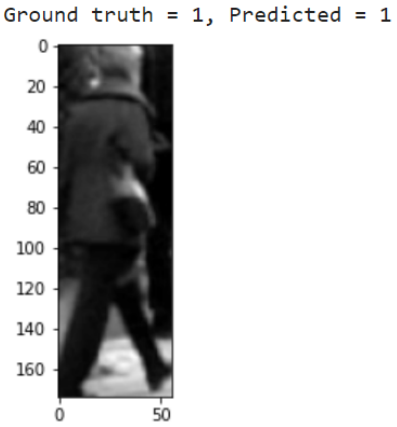
While precision refers to the percentage of the results that are relevant, recall refers to the percentage of total relevant results correctly classified by the algorithm. Precision and recall work better than accuracy if false positives and false negatives have different costs.

Both the precision and recall are consistent with accuracy. Since all of the evaluators are consistent, we consider all of them to be good representations of the dataset.

3.5 Sample Results

Figure 3.5.1 to figure 3.5.10 are the predicted labels for 5 person images and 5 non-person images, respectively.





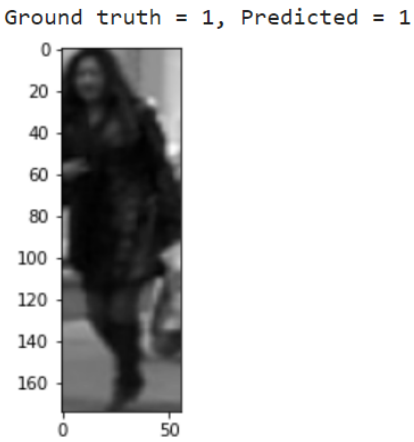


Figure 3.5.1 - Figure 3.5.5: Sample test results of person images

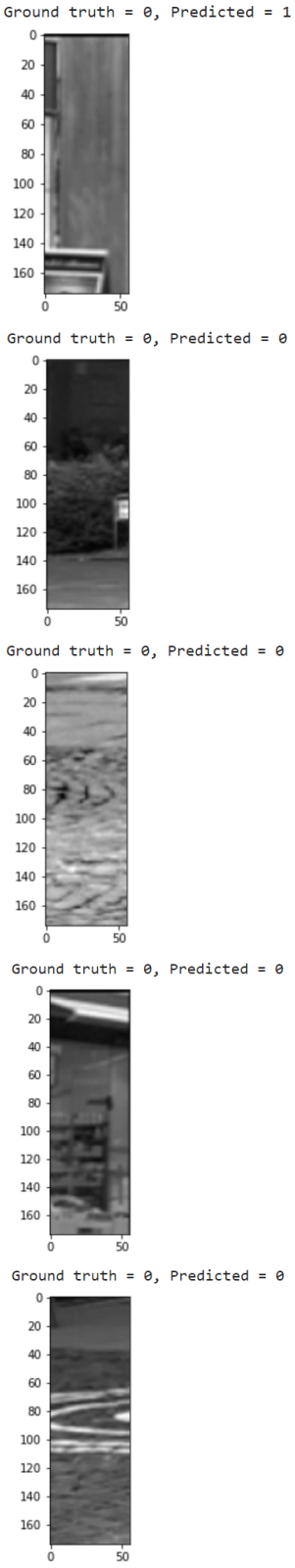


Figure 3.5.6 - Figure 3.5.10: Sample test results of non-person images

4. Detection and Localization

4.1 Description of the contents of the dataset

In this section, we have a total of 250 images for detection and localization, and the final number of identified “person” is 1042.

The final size of the bounding box is set to (250, 100). Moreover, the sliding window step size is 90 horizontally and 100 vertically.

4.2 Detection and localization method

The main objective of this part is to locate every “person” bounding box using the previously trained classifier. In our detection and localization algorithm, we are essentially distinguishing and drawing the person bounding boxes in each frame/image.

We start by determining the size of the sliding window. Our initial approach is to use the mean size of all bounding boxes as the size for the sliding window.

Furthermore, we start the algorithm by iterating the sliding window through the image and, in the meantime, call the previously trained classifier. If the binary classifier identifies the content inside the current location of the sliding window as a person, we would consider that there is a person element at that location.

Last but not the least, if the content inside the current location of the sliding window is identified as a person, we would record the position of the bounding box, draw a rectangle at that location and give it a distinct label.

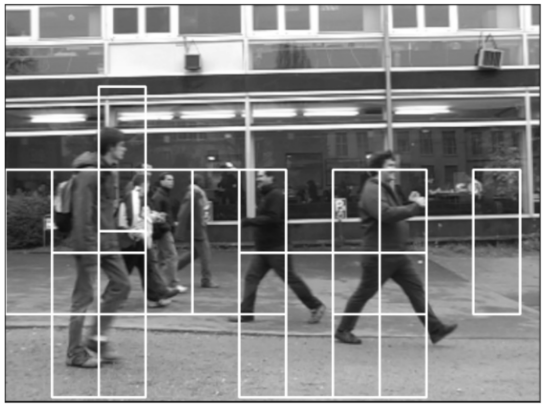


Figure 4.1: Detection and localization result containing overlapped bounding boxes

During the development process, we confront a few problems along the way. The first problem we had is, the classifier has identified and drawn too many person bounding boxes. As shown in Figure 4.1, a vast majority of the bounding boxes are overlapped and we could hardly distinguish between different people. To address this problem, we decide to merge the overlapping bounding boxes that refer to the same person. The result can be seen in Figure 4.2.

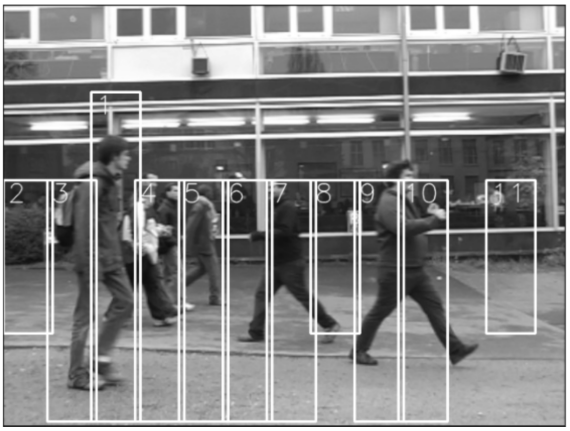


Figure 4.2: Detection and localization result after merging the overlapping bounding boxes vertically

As shown in Figure 4.2, even though some of these merged boxes are still pointing to the same person, the merged bounding boxes have successfully combined vertically distributed elements. Since combining the bounding boxes horizontally could result in a large bounding box that contains multiple people, we decide not to combine the horizontally overlapping bounding boxes. Thus, we understand that the optimal sliding window size is not necessarily the mean dimensions of the bounding boxes, but a larger size that could cover the majority of the person's size.

A better approach would be increasing the size of a bounding box. To make sure the new sliding window could fit into the classifier, we could also downsize the image in the first place.

4.3 Evaluation of detection, localization performance, and interpretation of results

By comparing the detected bounding boxes and the ground truth, we could see the IoU mean is 0.325 and Standard Deviation is 0.16 (see Figure 4.3 for more details).

```
IoU Mean: 0.3250013078245114
IoU Standard Deviation: 0.1624152543317621
```

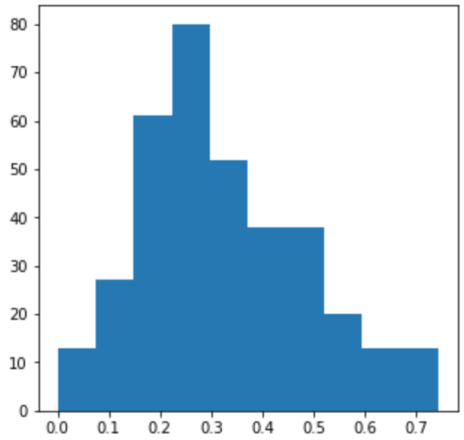


Figure 4.3: Mean and Standard deviation for IoU

5. Tracking

5.1 Tracking Method

In tracking, our goal is to find an object in the current frame, given that we have successfully tracked the object in all (or nearly all) previous frames. The idea mainly consists of using the first frame to detect the target, set the initial target frame, and finally call the tracking function. The tracking function can either be the Optical Flow tracker we implemented (see Figure 5.1.1) or the MOSSE tracker from OpenCV's built-in object tracker.

5.1.1 Optical Flow

The optical flow is a pattern of objects' motion [1]. In our case here, an optical flow of the video images could be the motion vector of each image pixel at different times. In addition, we are expecting those moving objects, which in our case refers to the people, to contain greater motion vectors, whereas the background pixels may stay steady.

The algorithm would start by iterating every image, and at the same time, recall its past image. Additionally, we would calculate the optical flow from these two images and reconstruct the motion vectors on pixels to the motion vector on selected bounding boxes.

An example we pick for the project is shown in Figure 5.1.1, which demonstrates optical flows on the pixels. There are in total of 480*640 motion vectors (dx, dy).

Therefore, among these 480*640 motion vectors, we would choose those vectors in the selected bounding boxes.

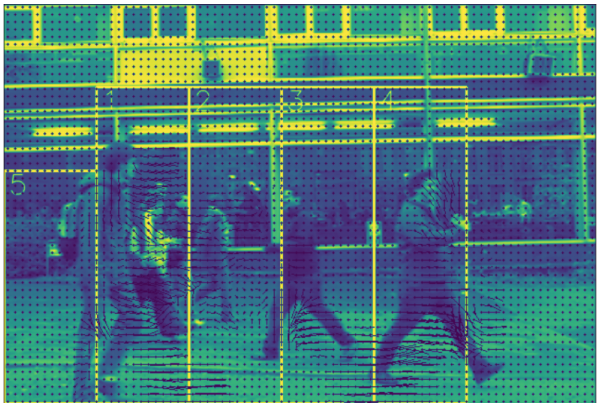


Figure 5.1.1: Optical flow on the image pixels

In the previous section, we record the position of the person bounding boxes. We would then combine and take the average of all the pixel-scale motion vectors into bounding box-scale motion vectors. so that each bounding box has its corresponding motion vector. Therefore, given the current bounding box and the motion vector, we could easily figure out the expected bounding box position in the next image. Lastly, from all candidate person bounding boxes from the next image, we would choose the closest one as the expected bounding box in terms of the IoU value.

5.1.2 MOSSE Tracker

Before using the MOSSE tracker, we created three trackers that would be used for the three individuals and initialized the three bounding boxes that each contained a chosen object/individual in the first frame. Then, we initialize the trackers with the first frame and the bounding boxes. Finally, we read frames from the images and just update the tracker in a loop to obtain a new bounding box location for the current frame. The results for each iteration are displayed.

We choose to use the MOOSE tracker mainly because it uses an adaptive correlation for object tracking [2], which produces stable correlation filters when initialized using a single frame [2]. It is not only robust to variations in lighting, scale, pose, and non-rigid deformations, but also capable of detecting occlusion based on the peak-to-sidelobe ratio, which enables the tracker to pause and resume where it left off when the object reappears [2]. Moreover, the MOSSE tracker tracks fast and operates at a high FPS [2].

Compared to other available OpenCV's built-in object trackers, the MOSSE tracker performs the best when paired with our trained classifier.

5.2 Tracking Performance Evaluation

In this part, we compute IoU to evaluate the tracking performance. Based on the equation provided in the project manual, in the numerator we compute the area of overlap between the predicted bounding box and the ground-truth bounding box.The denominator is the area of union, or simply, the area encompassed by both the predicted bounding box and the ground-truth bounding box. Dividing the area of overlap by the area of union yields our final result.

5.2.1 Optical Flow

Based on the IoU computation, we get a mean value of around 0.08619 and a standard deviation of around 0.138. Apparently, compared to the MOSSE tracker, it is not as accurate. In the video generated, we can see that the bounding boxes are not as active and accurate. Figure 5.2.1 illustrates the IoU distribution for Optical Flow.

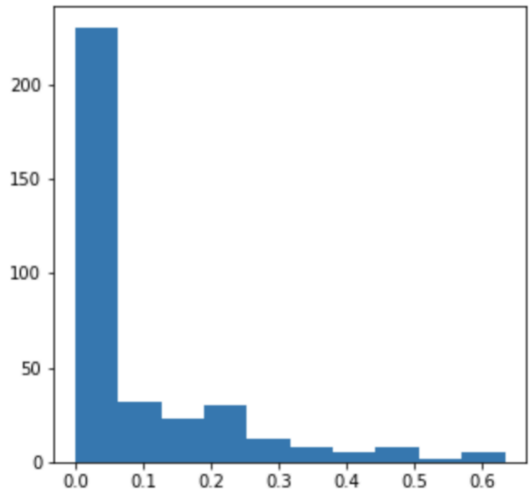


Figure 5.2.1: IoU distribution for Optical Flow

5.2.2 MOSSE Tracker

Based on the IoU computation, we get a mean value of around 0.23826 and a standard deviation of around 0.217. The MOSSE tracker produces better results in terms of the IoU. In the generated video, there are three instances of "person" successfully localized in the initial frame and those instances are tracked until they leave the frame. Figure 5.2.2 illustrates the IoU distribution for the MOSSE Tracker.

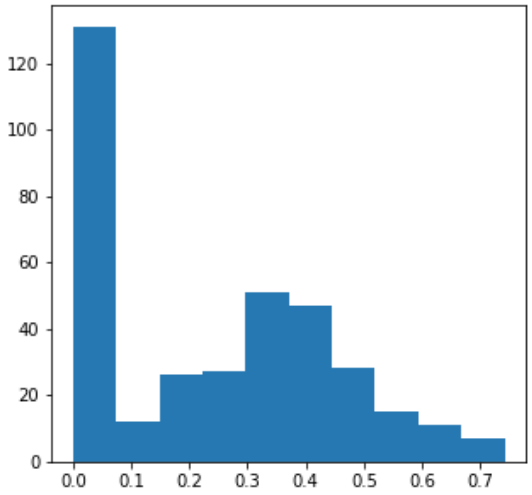


Figure 5.2.2: IoU distribution for the MOSSE Tracker

6. Reference

[1] Wikipedia, "Optical_flow." Available at https://en.wikipedia.org/wiki/Optical_flow#:~:text=Optical%20flow%20or%20optic%20flow,brightness%20pattern%20in%20an%20image.

[2] Tal Arbel, S 2021, "ECSE-415 Lecture-21 Tracker" lecture notes. McGill University, delivered 29th March.