Detecting, Localizing, and Tracking People

Lara Kollokian
McGill University
260806317

Rayan Osseiran
McGill University
260803963

Julien Philips
McGill University
260804197

Benji Szwimer
McGill University
260804222

Marie Vu
McGill University
260807903

Abstract

The purpose of this project was to implement a pipeline capable of detecting, localizing and tracking individuals walking across a scene from a fixed viewpoint. The dataset consists of frames from the campus and Stadtmitte videos. Classification was done by extracting local binary patterns from a subject, then training a model using an SVM classifier. The mean accuracy for classification was 99.6% after 5-fold cross-validation. Localization was done through the use of a sliding window and non-maximum suppression and a mean IoU of 15.2% was reported across validation sets. Finally, two tracking methods were used: optical flow and kernelized correlation filter (KCF). Optical flow was tested on the campus and Stadtmitte training videos with and without relocalization. The mean IoU for the campus video with relocalization was 0.39%, and without relocalization was 0.46%. The mean IoU for the Stadtmitte video with relocalization was 0.23%, and without relocalization was 0.52%. Similarly, the KCF algorithm was tested on the campus and Stadtmitte training videos. The mean IoU for the campus video was 0.35% while the mean IoU for the Stadtmitte video was 0%.

1. Introduction

Human detection is an interesting task because it builds a foundation for many complex tasks that are possible with the aid of computer vision. From autonomous vehicles, security, to sports analytics, the concepts discussed in this project build the foundation for many interesting areas of this quickly expanding field. In order to perform all the tasks set out for this project, it was necessary to divide it into four tasks: data extraction, classification, detection, localization and tracking. The following methods were implemented: In order to perform classification, we used local binary patterns (LBP) and a support vector machine (SVM) classifier. Once we had a working classifier, we then went on to do detection and localization. The detection and localization portion of our pipeline encompassed using a variable size sliding window in order to detect people in the images as well as non-maximum suppression in-order to select the best bounding boxes. Finally, to complete the tracking portion of our pipeline we took advantage of both optical flow and KCF.

2. Data

The main purpose of the data portion of this project is to acquire and format the data so it can be used for classification purposes. The data provided in this case includes the frames from multiple videos as well as the ground truth for those frames. The ground truth indicates the bounding boxes that surround the people in each frame. Two videos were used for training purposes, TUD-Campus and TUD-Stadtmitte.

The data required for classification was the training data as well as the training labels. Training data is essentially a list of person and non-person image patches extracted from the video frames. Labels are a list of zeros and ones for each of the non-person and person patches respectively. An additional consideration in extracting the data was making sure the data could be easily separable with respect to the people in each video. In order to produce correct cross validation accuracies in the classification, the same people must either be in the training set or the validation set. The reason data must be split according to the people in each video is to ensure that the validation accuracy reflects the performance of the classifier on unseen data.

The approach taken to get the data began with extracting the person patches from all the frames. This involved parsing the ground truth files to get a dictionary of frame keys mapped to a list of bounding boxes for that frame. We then iterated through all the frames and extracted the person patches according to the bounding boxes. It is worth noting that some bounding boxes had negative coordinates or coordinates that exceeded the image size, so we adjusted these bounding boxes so that they fell within the frame's bounds. Extracting non-person patches was more complex. A sliding window approach was used to extract the non-person patches. The sliding window used was a random size that fell within the range of the largest and smallest dimensions of the bounding boxes in that frame. At each iteration of the sliding window, we evaluated whether the window intersected with any of the bounding boxes. In the case that it did not intersect with any bounding boxes, it was successfully added to a list of non-person patches. Generating labels for the data was also trivial given that we extracted two separate lists of person patches and non-person patches.

The final consideration for the data was to be able to split the data for cross validation in classification. The approach taken to do this was to remap the ID of each person in each video to an ID that encapsulates both the person's ID as well as the video it came from. In addition, each non-person patch was assigned a unique ID. Doing this allowed us to have a list of IDs for all the person and non-person patches. This list of IDs is later passed into sklearn's `GroupKFold` as the `groups` parameter, ensuring that the same person is not in both the training and validation sets.

The final result of the data section is the training data, training labels and the corresponding ids for all the data. The training data consists of approximately four thousand samples, with forty percent of that data being person patches and the other sixty percent being nonperson patches.

3. Classification

Many approaches were explored when building the human classifier. Initially, the classifier would extract SIFT descriptors, which would then be clustered via k-means or expectation-maximization. Once clustered, a histogram is made using bag of words representation and is then fed to a classifier. Whilst this approach should have produced good results in theory, the true results were far from perfect. Indeed, the accuracy ranged from as low as 20% to as high as 73% which depended on how the dataset was generated.

To that effect, we attempted to explore other approaches. The next contender was histogram of oriented gradients (HoG). One of the primary advantages of this approach was that it would reduce the number of intermediary steps as it directly produces a feature vector that can be passed to a classifier. Unfortunately, when testing on the original full-sized images, Google Colab would crash due to a lack of memory. Indeed, storing thousands of large histograms takes up space very quickly. Moreover, assuming we did manage to get around this issue, we would still likely need to use principal component analysis (PCA) in-order to reduce the dimensions of the feature vectors. Finally, HoG would also require the images to be padded which is something that we do not do.

This left us with our final approach which takes advantage of local binary patterns. Like HoG, it produces a feature vector that can directly be fed to a classifier. Moreover, while it is memory hungry, it requires that images are converted to grayscale which likely heavily reduces the dimensionality. Images do not need to be resized for LBP and they can also be left as the same size, as the histogram will always be the same size regardless of the image size.

In addition to an SVM based classifier, we also made use of gradient boosting as an alternative. One aspect of gradient boosting that we were particularly attracted to was gradient descent, as it can be a very powerful optimization tool. Moreover, it is very accurate out of the box with limited tuning. In general, it is robust to overfitting compared to many other classifiers, however, attention still needs to be paid when optimizing the hyperparameters as the number of estimators needs to be kept in check [1].

3.1. SVM Results

The SVM approach produces reliably high accuracy and the results from cross validation are listed in Table 1. As noted in the data section, cross-validation was done through the use of Sklearn's `GroupKFold` which allows us to ensure that no subjects in the validation set appear in the training set. It is encouraging to see a high precision, as precision tells us what portion of our results are correct.

Metric	Value (%)
Mean Accuracy	99.6
Standard Deviation	0.2
Precision	99.6
Recall	99.6

Table 1: Cross-validation results for SVM

Parameters for SVM as well as LBP were selected via grid-search (a custom implementation). The optimal parameters are highlighted in Table 2. Note that the number of points in LBP is not optimized via grid search due to the exponential increase in runtime when adding a new hyperparameter. Instead, the number of points is directly based on the radius. The decided formula is 8 * radius, primarily because many sources display a radius of 1 and 8 points as a standard configuration [2][3]. Moreover, we observe that both typically scale together hence allowing one to be inferred from the other.

Parameter	Value
C (SVM)	1.0
Kernel (SVM)	rbf
Radius (LBP)	1.0
Number of Points (LBP)	8
Method (LBP)	default

Table 2: Optimal SVM parameters selected by grid search.

Note that for grid-search, we do not simply select the option with the highest mean accuracy, but instead try to select a combination that has high accuracy, but with parameters that seem less likely to overfit. As an example, the top candidate for SVM had a regularization parameter (C) of 10. We opted to select the next option with a regularization parameter of 1 for safety reasons. A sample run of the classifier is shown in Figures 1 and 2. Figure 1 shows 5 patches that the classifier predicted as human, whilst Figure 2 shows 5 patches that the classifier predicted as non-human. In each patch the number above is the ground truth. 1 indicates human whilst 0 indicates non-human.

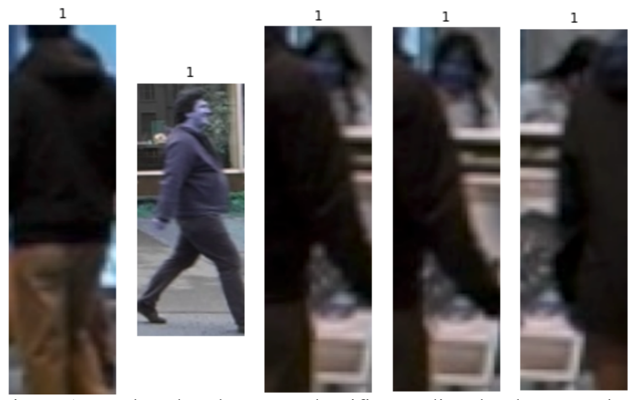

Figure 1: Patches that the SVM classifier predicted as human. The number above each patch is the ground truth.

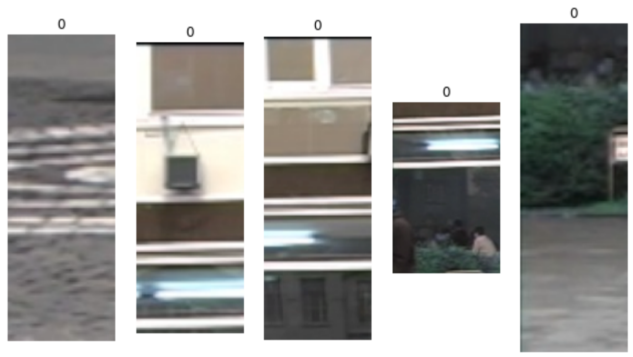

Figure 2: Patches that the SVM classifier predicted as non-human. The number above each patch is the ground truth.

3.2. Gradient Boosting Results

Gradient boosting (GB) performed very similarly to SVM. We suspect that this is the case because we selected a non-linear kernel for SVM. Had we selected a linear kernel, we would expect gradient boosting to do a better job. Once again, the classifier was evaluated with grid-search and 5-fold cross-validation. The results of cross-validation are illustrated in Table 3 and the optimal hyperparameters from grid-search are listed in Table 4.

Metric	Value (%)
Mean Accuracy	99.5
Standard Deviation	0.7
Precision	99.9
Recall	99.9

Table 3: Cross-validation results for gradient boosting

Parameter	Value
Learning rate (GB)	0.1
Number of estimators (GB)	100
Radius (LBP)	1.5
Number of Points (LBP)	12
Method (LBP)	default

Table 4: Optimal gradient boosting parameters selected by grid search.

A sample run of the classifier is shown in Figures 3 and 4. Figure 3 shows 5 patches that the classifier predicted as human, whilst Figure 4 shows 5 patches that the classifier predicted as non-human.

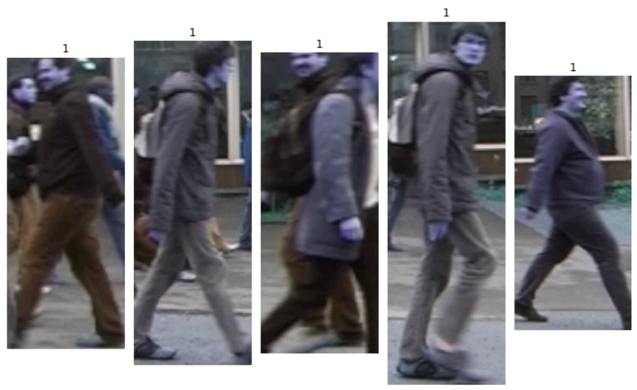

Figure 3: Patches that the gradient boosting classifier predicted as human. The number above each patch is the ground truth.

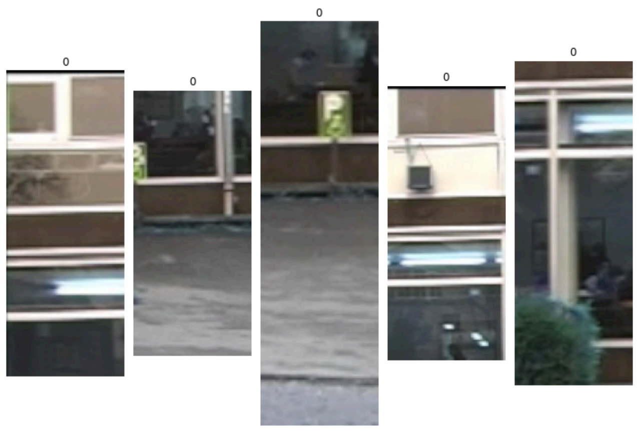

Figure 4: Patches that the gradient boosting classifier predicted as non-human. The number above each patch is the ground truth.

3.3. Weaknesses

Unfortunately, this classifier is not without its weaknesses. While the dataset after processing contains roughly 3900 elements, it is important to realize that all of these patches come from the same 2 videos, meaning they consist of the same or similar people over and over again. More importantly, this means that it consists of the same overall environment and weather conditions. Unfortunately, simply adding more videos does not necessarily produce the result we'd expect. This can actually harm performance in localization. Moreover, it is easy to accidentally overfit to non-person patches. Indeed, the classifier could fail to properly classify foreign environments that are completely different from its training environment. A good example of this is the Sunnyday dataset where the time of day is dusk and there are many glass window reflections. To that effect, the main point here is that it is important to look at the context the classifier is being used in i.e., in this case localization and tracking. Moreover, it would be worth looking into combining LBP and HoG in the future as many papers cite it as a very strong feature detector for pedestrian detection [4][5]. This is not too surprising as we feel that HoG is better than LBP at getting the overall shape of an object, whereas LBP is fully focused on textures.

4. Localization

In order to perform detection and localization we decided to go with the following approach: In order to localize people in the frame, we used a sliding window method in which different window sizes were chosen. We then iterated through the image with these sliding-windows and at each iteration, we used our classifier to determine whether or not a person was present in the window. If a person is detected, we store the patch in a list. Once the entire image is iterated through with these varying sliding window sizes, we then apply non-maximum suppression on the bounding boxes that were found from this sliding window approach. Non maximum suppression allowed us to select a single bounding box for the people out of the many overlapping bounding boxes that were detected.

In localization, the dataset consists of frames, instead of patches like in classification. For the purpose of localization, we used the training videos (which consists of 250 frames), however, we worked to ensure that the IoU was not calculated using any subjects that the classifier was trained on. This is a rather complex process. Briefly, all bounding boxes in the ground truth are checked. The image from the bounding box is then reconstructed. If that patch exists in the train set, we store it so that we can remove its matching predicted bounding box later. Otherwise, we will use this for calculating the IoU. Once we know the images of interest in the ground truth, we also need to discard the frames from our localization prediction that are predicting people from the training set. To do this, we iterate over our predicted bounding boxes for each frame. For each box, we compare them one by one to the boxes in the previously stored ground truth that contains patches from the training set. If the IoU between the two boxes is over 0.5, we can safely conclude that the prediction box is predicting a person from the training set. Therefore, we discard the box from predictions. Finally, we can move on to calculating the IoU based only on subjects from the validation set of the classifier. Note that false positives are left alone and will be given an IoU of 0.

4.1. Results

Localization results were once again validated using `GroupKFold`. While it makes limited sense to do cross validation here because we have already trained our model, the primary benefit is that it handles splitting into 5 validation sets. With that being said, it’s worth noting that our split does not always successfully get an IoU for all five folds. This is intentional behaviour to prevent errors. Suppose all the images in the validation split contain subjects in the classifier’s training set. We cannot accurately predict an IoU without including training boxes, so discard it. Moreover, we perform grid-search in-order to select the best IoU threshold for non-maximum suppression. Briefly, that decides what the IoU cut-off is for removing a frame. Indeed, this is our evaluation metric for non-maximum suppression as our predictions do not have confidence scores attached with them. The threshold that was selected via grid search is 10%. The results for the validation are reported in Table 5.

Metric	Value (%)
Mean IoU	15.4%
Standard Deviation	9.4%

Table 5: Optimal gradient boosting parameters selected by grid search.

One potential explanation for the high standard deviation is that we are not counting boxes that include subjects the classifier was trained on. Suppose in that case that we have a frame with 4 people and that 3 were in the training set. If we have a bounding box over the other person, as well as a false positive, that is going to give us a far lower IoU than considering all boxes. Figure 5 shows a sample of localization results.

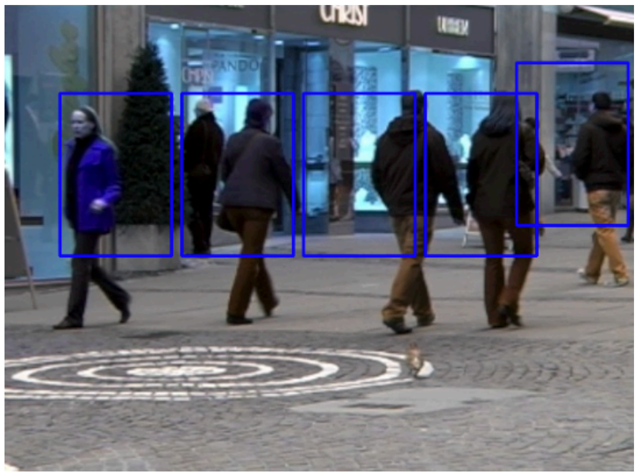

Figure 5: A sample localization result.

4.2. Weaknesses

As the previous section may indicate, IoU in localization is not particularly good. However, as we can see in Figure 5, the results are not too bad from a human perspective. With that being said, IoU places a heavy emphasis on getting the bounding boxes perfectly and our model with 3 fixed bounding box sizes is never going to accurately achieve that (at least in its current configuration). Moreover, this also points to issues with our classifier. Indeed, because LBP is a texture-based feature detector, we see many instances where half a person, or a person's leg is considered a person. This points to a need to better capture the overall shape of our classes instead of just textures. Finally, one localization issue is also that the classifier will detect every single person, whether far or close. We don't necessarily want to place bounding boxes on someone very far in the background. This is a non-trivial issue to solve as we would likely need to use stereo vision techniques in-order to understand the depth of objects.

5. Tracking

Tracking is a real challenge in computer vision, because of things like occlusion, rotation, shadows, drift and objects that appear or disappear from the frame [6], which is why this portion of the pipeline was so interesting. The tracking step starts off with localization. We use our localization algorithm to predict the bounding boxes for the first frame in a video and use tracking to follow the moving objects in the video. This can be done strictly with tracking, or by relocalizing every few frames to detect new images that appear. We tested out two different tracking methods: optical flow and KCF.

5.1. Optical Flow

The method we implemented for Optical Flow tracking was using the Shi-Tomasi Corner detector to find good features to track in a frame, and then the Lucas-Kanade Optical Flow to track them from frame to frame.

The Shi-Tomasi feature tracker uses the eigenvalues of the second moment matrix to find good features to track in a frame. According to Shi-Tomasi, good features are the ones whose motion can be estimated reliably [7]. The OpenCV implementation of the algorithm, `cv2.goodFeaturesToTrack()`, allows you to input a mask to specify the region of the image where the corners should be detected. We decided to create a mask in the shape of each frame where the specified region is the area within the bounding box for that frame. You can see in Figure 6 and 7 below what that would look like. Figure 6 generates a mask that would look like Figure 7.

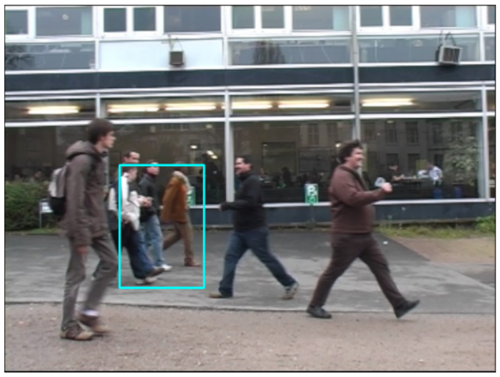
Figure 6: A frame of the campus video with one bounding box.

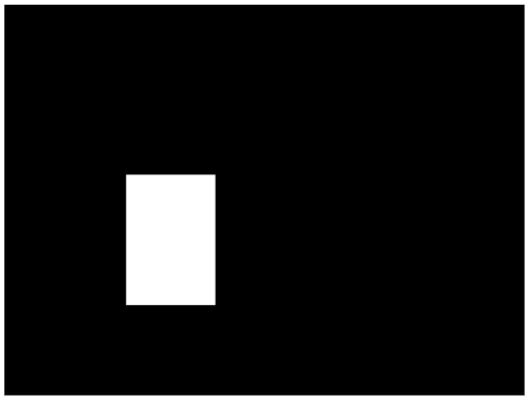
Figure 7: Mask corresponding to the frame in Figure 6.

After running Shi-Tomasi corner detection on a frame, we used Lucas-Kanade optical flow to calculate the updated positions of the tracked corners from the present frame to the next frame. Lucas-Kanade works by warping image patches around the tracked corners to find the transformation between two frames [7]. This essentially gave us two sets of mapped points from the present frame to the next frame. In order to calculate the position of the new bounding box, we calculated the average displacement of all the points from the present frame to the next frame. This gave us an average displacement in x and in y for the features within that bounding box. We then applied that transformation to the bounding box to get the new coordinates for the next frame.

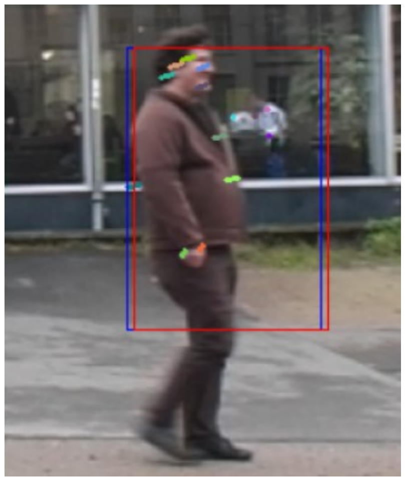
Figure 8: Example of the displacement of points between two frames, as well as the previous and next bounding boxes

In Figure 8 above, you can see the displacement of the points detected by Shi-Tomasi and Lucas Kanade demonstrated by the coloured lines. You can also see the previous bounding box in blue, and the next bounding box in red.

We also incorporated relocalization into our tracking method. We wanted to make sure to catch new objects that appear within the video, and also correct any accumulated error from the tracking. We experimented by trial and error to find a good interval of frames after which to relocalize and we found that 20 was a good value. If we relocalized more often than this, the video looked very choppy and there was no smooth motion of the bounding boxes between frames.

5.2. Optical Flow Results

Given that validation was done using 5-fold cross validation, we didn't have a specific validation set for the training videos. Therefore, the mean IoU was computed over the entire campus and Stadtmitte training videos, both with and without relocalizing every 20 frames.

The mean IoU for the campus video with relocalization every 20 frames was 0.0039, and without relocalization was 0.0046. Figure 9 is a plot of the IoU per frame of the video both with and without localization

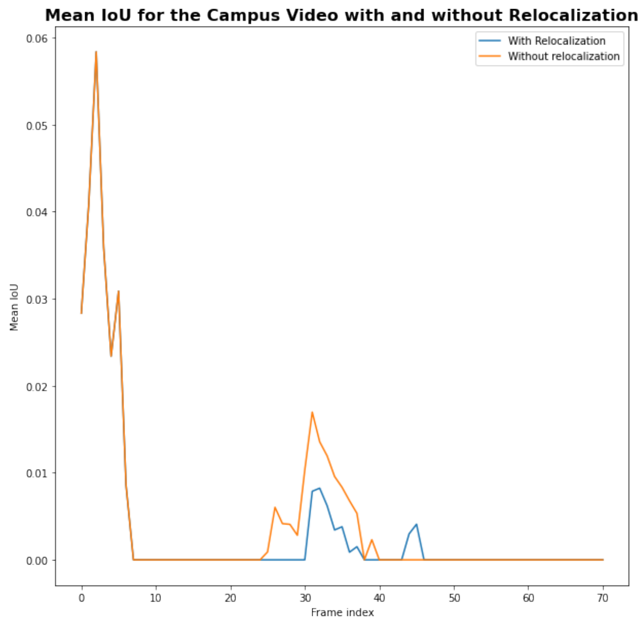

Figure 9: Plot of the mean IoU per frame of the campus video with and without relocalization.

The mean IoU for the Stadtmitte video with relocalization every 20 frames was 0.0023, and without relocalization was 0.0052. Figure 10 is a plot of the IoU per frame of the video both with and without localization.

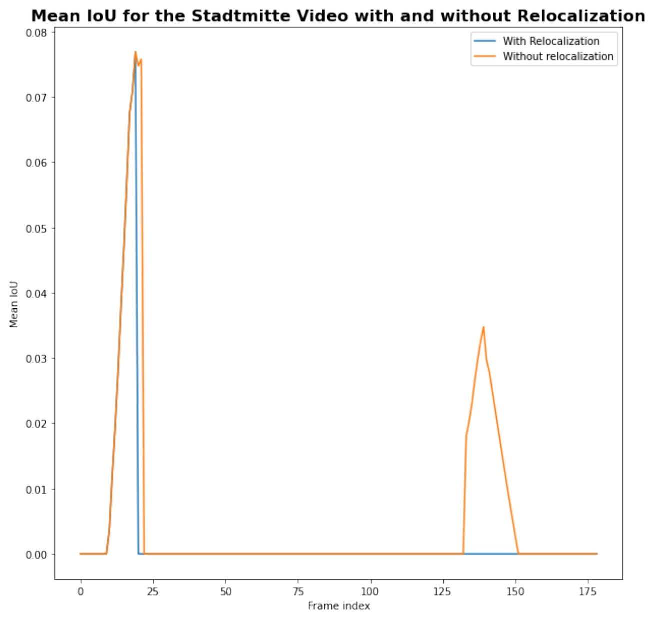

Figure 10: Plot of the mean IoU per frame of the Stadtmitte video with and without relocalization.

The IoU results for both videos are much lower than expected, and we're not too sure why. A part of it is likely because the size of the bounding boxes we predict don't match the size of the ground truth bounding boxes, which introduces a lot of error. Although the IoU numbers aren't too promising, the visual results look really good. You can see by viewing the video titled "`optical_flow_tracking.mp4`" included with the submission.

5.3. KCF Tracking

OpenCV offers 8 different tracking algorithms that can be used to track any desired object. We decided to test out the KCF tracker, which stands for kernelized correlation filter. This algorithm claims to be fast but doesn't handle occlusion well. This uses kernel trick and circulant matrices to significantly improve the computation speed [8].

To handle tracking multiple objects, `cv2.MultiTracker_create()` was used. This class is used to track multiple objects using a specified tracker algorithm.

To start, we first localize and detect the first frame that forms the video. This gives us the initial positions of the people located on the video. From there, we iterate through these bounding boxes' locations and add them to the MultiTracker.

From there, we go through all of the image frames and use update to update the tracking status of the current frame. We also append all of these new coordinates to an array so that we can display the bounding boxes on the frames, which is done by iterating through all the frames and drawing boxes with `cv2.rectangle` on each frame.

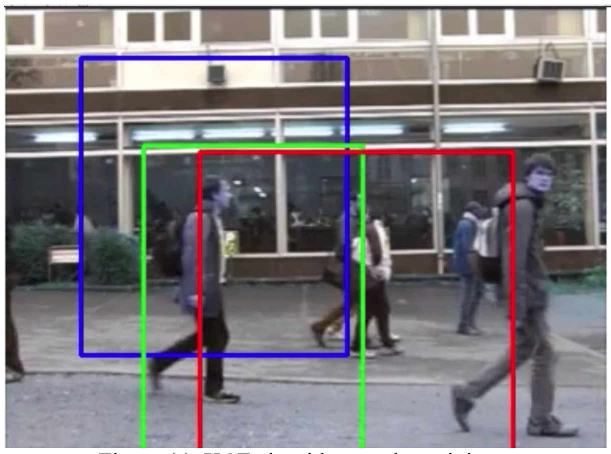
Figure 11: KCF algorithm on the training set.

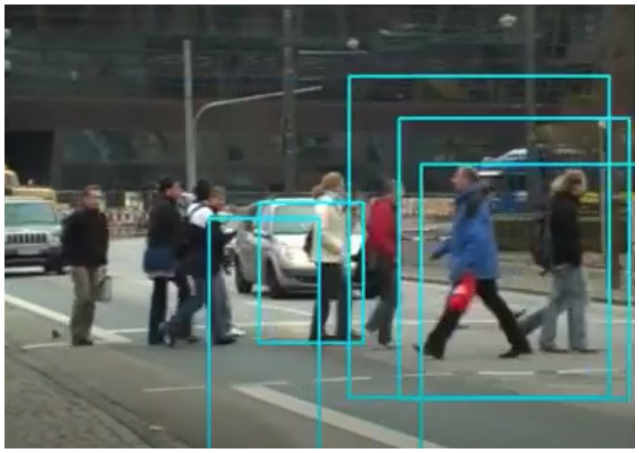
Figure 12: KCF algorithm on the test set.

Figures 11 and 12 shows that the algorithm is not super accurate. Some boxes show false negatives, and some people are also missing. Additionally, the sizes of the bounding boxes are too big compared to the size of the person detected.

5.4. KCF Results

Similarly, to Optical flow, we didn't have a specific validation set for the training videos and computed the mean IoU over the entire campus and Stadtmitte training videos.

The mean IoU for the campus video using the KCF algorithm was 0.003465 and the plot of the mean IoU per frame of the video is shown in Figure 13.

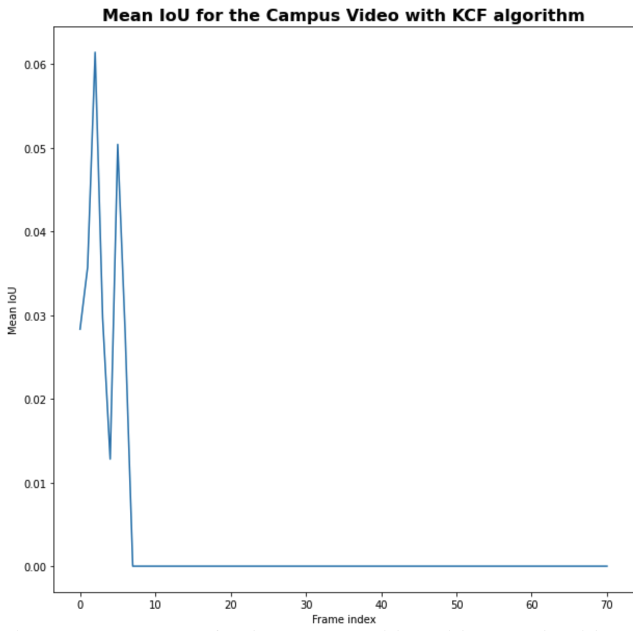

Figure 13: Mean IoU for the Campus Video with KCF algorithm.

The mean IoU for the campus video using the KCF algorithm was 0.0 and the plot of the mean IoU per frame of the video is shown in Figure 14.

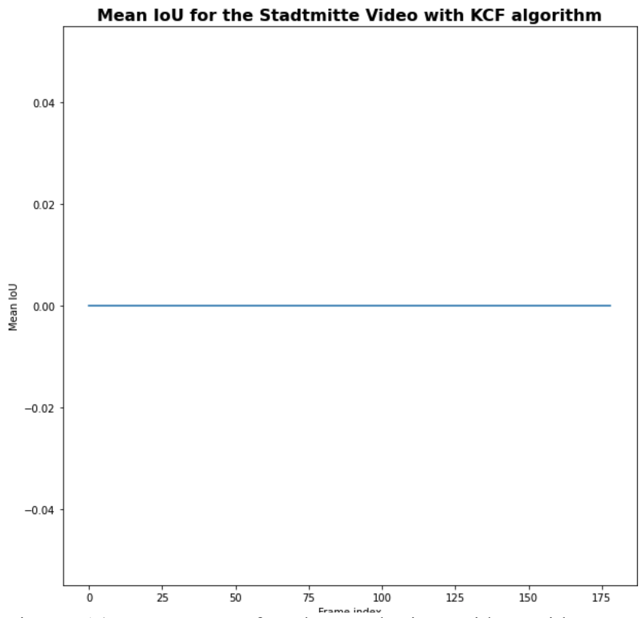

Figure 14: Mean IoU for the Stadtmitte Video with KCF algorithm.

As observed, the results of IoU are very low. Looking at our videos for the KCF, it is pretty expected. However, it is interesting that the mean IoU for Stadtmitte is 0.0 as there seemed to be at least one detection when we tested the algorithm on that set. The large size of our bounding boxes would have influenced the result as they aren't close to those of the ground truth bounding boxes. The results can be viewed in the following: "`kcf_tracking.mp4`".

6. Conclusion

The objective of this project was to implement a complete pipeline to detect, localize, and track individuals walking across a scene from a fixed viewpoint. We implemented classification, localization and tracking functions using data from the campus and Stadtmitte videos. Different methods were tested, and hyperparameters were tuned through grid search and cross-validation. Our results were promising overall, but the performance metrics weren't too optimistic for some of the steps. As previously mentioned, attempting to combine HoG and LBP would be a promising change to implement for the future. This would combine LBP's strengths with textures with HoGs ability to capture shapes as well as textures well. It would also help balance out the accuracy of our classifier over person and non-peron patches. Having a more accurate classifier would also improve the performance of the rest of the pipeline. Another potential improvement would be to consider dynamically resizing bounding boxes around an object while localizing and tracking, rather than using the three fixed sizes we currently have in our implementation. This would improve the IoU results we obtained for tracking. This was a very large bottleneck in localization and represents a challenging problem to solve. Finally, a simple fix to consider as well would be to train on more data with a fixed reference, which the extra videos do not have. This would give us more varied environments to work with which reduces the chances of the classifier overfitting on a single environment.

References

[1] G. Snow, "Why is boosting less likely to overfit?," 24-Jan-2017. [Online]. Available: https://stats.stackexchange.com/questions/257328/why-is-boosting-less-likely-to-overfit/257921#257921.

[2] "Local Binary Pattern for texture classification," *scikit-image*. [Online]. Available: https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_local_binary_pattern.html.

[3] A. Rosebrock, "Local Binary Patterns with Python & OpenCV," *PyImageSearch*, 07-Dec-2015. [Online]. Available: https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/.

[4] W. Park, D. Kim, Suryanto, C. Lyuh, T. M. Roh and S. Ko, "Fast human detection using selective block-based HOG-LBP," *2012 19th IEEE International Conference on Image Processing*, Orlando, FL, USA, 2012, pp. 601-604, doi: 10.1109/ICIP.2012.6466931.

[5] G. Gan and J. Cheng, "Pedestrian Detection Based on HOG-LBP Feature," *2011 Seventh International Conference on Computational Intelligence and Security*, Sanya, China, 2011, pp. 1184-1187, doi: 10.1109/CIS.2011.262.

[6] T. Arbel, Class Lecture, Topic: "Lecture 19 - Motion Analysis." ECSE 415, Faculty of Engineering, McGill University, Montreal, QC, Mar. 2021.

[7] T. Arbel, Class Lecture, Topic: "Lecture 21 - Tracking." ECSE 415, Faculty of Engineering, McGill University, Montreal, QC, Mar. 2021.

[8] S. Yadav and S. Payandeh, "Understanding Tracking Methodology of Kernelized Correlation Filter," *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Vancouver, BC, Canada, 2018, pp. 1330-1336, doi: 10.1109/IEMCON.2018.8614990.