Pedestrian Tracking with Combination of YOLOv4 and CSR-DCF

Andrew Wei awei@umich.edu
Wei-Chun Lu wclu@umich.edu
Leo Lee leowcl@umich.edu
Geric Pan gericpan@umich.edu

Abstract

This paper reports the detecting and tracking of the pedestrians in the recorded video from a camera installed on a driving car. You Only Look Once (YOLO), detecting the custom objects by training with the Caltech Pedestrian Dataset, was utilized for detecting pedestrians in initial frames. The tracking of the pedestrians was conducted by the tracker based on the Discriminative Correlation Filters with Channel and Spatial Reliability (CSR-DCF). The tracking results and the evaluations of the performance of the detector and tracker are provided with experiments.

1. Introduction

In recent years, many advancements were made in the development of autonomous vehicles. Some vehicles today have the ability to automatically brake if it detects an upcoming collision, or safely and reliably drive on the highway. However, there are still many obstacles to tackle before we can make use of vehicles that are fully autonomous. One such obstacle is pedestrian tracking. In order to safely operate on the road, autonomous vehicles must consistently detect and track the motion of pedestrians and make decisions based on that motion. In our project, we aim to detect and track the motion of pedestrians from videos taken from a vehicle driving through traffic.

Reliable pedestrian tracking is one of the crucial problems that we must overcome to achieve fully autonomous vehicles. If we are able to develop systems that can track the motion of pedestrians with high degrees of efficiency and accuracy, we will be one step closer to making autonomous vehicles a reality. Moreover, the detection and tracking algorithms used to solve the problem of pedestrian tracking can also be applied to solve many other problems in computer vision, such as surveillance, medical imaging, and augmented reality.

At its core, the problem of object tracking is deep learning problem. To fully solve this problem would be to find the optimal pedestrian detection algorithm in terms of efficiency and accuracy. While this ideal solution may still be out of reach, the problem of detecting and tracking pedestrians is achievable, and there already exists several object detection and object tracking algorithms that can produce decent results. Some of the common object detection algorithms include Histogram of Oriented Gradient (HOG) and You Only Look Once (YOLO), which are based upon Support Vector Machine and deep learning approaches respectively. Many popular object tracking algorithms also use deep learning methods, such as Generic Object Tracking Using Regression Networks (GOTURN) and Multi-Domain Convolutional Neural Network Tracker (MDNet). Using similar detection and tracking algorithms, we should be able to solve the problem of pedestrian detection and tracking.

The problem of pedestrian detection and tracking has been a hot topic of research in recent years. In 2017, a group of computer scientists from Stanford University developed a YOLO-based adaptive window approach for detecting humans in videos, and demonstrated that YOLO was an efficient and effective method for detecting humans [3]. In 2018, a paper published for the IEEE International Conference on Mechatronics and Automation explored pedestrian detection, building upon the YOLO algorithm to modify the network structure. Results showed that this modified version of YOLO effectively improved the accuracy of pedestrian detection, while minimizing the false detection and missed detection rate [5]. As for pedestrian tracking, a paper published in the Journal of Advanced Transportation in 2018 described a solution to automatic multipedestrian tracking using real-time Tracking-Learning-Detection algorithms. This method was shown to be accurate and stable, with the hopes of improving the accuracy in high density situations [8].

In our project, we aim to detect and track the motion of pedestrians as seen through videos taken from a driving vehicle. To accomplish this, we will first use YOLOv4 (You Only Look Once) to detect and create bounding boxes around pedestrians in the first frame of the video, and then use Discriminative Correlation Filters with Channel and Spatial Reliability (CSR-DCF) to track the motion of the pedestrians.

2. Approach

In this paper, the framework of a multi-pedestrian tracker is constructed. Generally, the bounding box of pedestrians in the initial frame is given by a detection algorithm called YOLO, and then the pedestrians are tracked by the tracker based on the CSR-DCF method in the subsequent frames of video.

2.1. Object Detection

You Only Look Once version 4 (YOLOv4) applies a single neural network to a full image, that divides the image into regions and calculates probabilities and bounding boxes for each region. Looking at the whole image together allows YOLOv4 to be extremely fast, and also makes predictions as informed by the context of the entire image.

YOLOv4 was released by Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao [2] in April 2020 and claimed as one of the state-of-the-art real-time object detectors. The new architecture of YOLOV4 is built with CSPDarknet54 as a backbone, which promotes the learning capability of CNN. Furthermore, the implement of universal features includes Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT) and Mish-activation help YOLOv4 obtain a very impressive results. Referring to its paper, YOLOv4 is over 10% faster and more accurate compare to YOLOv3.

The dataset we used to perform pedestrian detection and tracking is a subset of the Caltech Pedestrian Dataset [1]. The dataset consists of videos taken from vehicles driving through traffic, and is comprised of 10 sets of .seq files, with each file representing a one minute long video. The dataset also includes annotations for the ground truth labels of pedestrian positions in each frame of every video sequence, which will be used for training our YOLO model. In our project, we decided to use only one set of videos (set07), where we separated videos 0-7 for training, and videos 8-11 for testing. Each video contained roughly 2000 frames, so we selected every 10th frame to include in our data. This was done to provide more variety between training data, since subsequent frames of a video are very similar. Even after this, we had over 1500 images to train with.

The format that that YOLO requires for training/testing data and labels is different from the format that was provided in the Caltech Pedestrian Dataset, so we found and modified code from a repository that converted the dataset files into a YOLO-readable format [9]. The code in the repository converted every single image in the dataset files into square images. Since we wanted the original dimensions and only some of the images, we modified the code to suit our needs. Using this dataset, we trianed our YOLO model and began making predictions for pedestrian bounding boxes in our test set. These bounding box values would later be provided as input for the tracking algorithm.

2.2. Object Tracking

For object tracking, Discriminative Correlation Filters with Channel and Spatial Reliability (CSR-DCF) [6] is used in our work. The spatial reliability map adjusts the filter support to the part of the target object, overcoming the problem of enlarging selected regions and the limitation of rectangular object assumption. The channel reliability contributes to weighting the responses of channel-wise filters in localization. CSR-DCF is an efficient method which leverages two features, HoGs and Colornames. It is included in the OpenCV tracking API, and thus, in this project, we build a multi-pedestrian tracker based on the CSR-DCF tracker API in OpenCV.

However, the CSR-DCF tracker itself cannot handle some situations. For example, it cannot stop tracking an object which exits the range of the camera or add new tracking targets which enter the visible region. Therefore, we adopt three strategies to overcome some problems we countered.

1. Stop tracking when the center of the bounding box is out of the image.

2. Read the detection information given by YOLO every n frames, where n is an user defined parameter. This helps adding new objects for tracking and deleting the objects that are no longer visible in current vision.

3. Check the bounding box and compare it to the original image. If they become too dissimilar we delete the bounding box and try to track it again after running YOLO.

The first on the list was a first priority for us. An issue that the tracker had at times was when object was no longer visible from the screen whether because it was blocked off by another object or left the image altogether, the bounding box would stubbornly stay on the screen and stay frozen in place or change course and follow another passing item. We decided the best way to handle this was to combine YOLO with object tracking, alternating between the two, so that when the tracking was going off course, YOLO would put it back on track. Currently a YOLO program is run which returns a text file containing the ID of objects and the location of their bounding boxes. The tracker reads in the YOLO text's output and uses it as the reference for the YOLO identification.

We decided the best value for n would be 20 frames. If we were to go below that, it may trivialize the work of the tracker and if we went above that it would be more likely for the tracker to accidentally track something we don't expect or want to. We would compare the contents of the bounding box of the first frame to what was in the bounding box in the current frame. We tried many methods including root

mean square error and SIFT but they tended to have problems of either the measured difference as a percentage was too narrow to properly separate or there wasn't any standard metric for that could be applied to all videos. We eventually decided on using the correlation coefficient as it had a better standard to measure against. By testing it out on many inputs we found that the approximate optimal value for the correlation coefficient was around 0.2, or if the coefficient were under 0.2, it is likely so different from the original image, in that it's been covered by something else or it's started to track an unrelated entity, that it would make more sense to not track it and let YOLO find it again once the n frames had passed to have a definite starting point and more accurate tracking. We also used the correlation coefficient when deciding to use the YOLO tracker. YOLO, while accurate could cause problems if it identifies a spot where the object actually is but is hidden. If we blindly accepted the YOLO's bounding box the following tracker's path would likely be very skewed from what it actually is, therefore we consult the coefficient and if it is under 0.2, we shouldn't try tracking it until a later YOLO output gives a better result, following the same logic as above.

The code displays the tracking and YOLO in realtime and also returns an mp4 file of the tracking for future viewing.

3. Experiments

3.1. Detection

Here we will discuss the quantitative results from out training of the YOLO model. Figures 1 and 2 show the results of our accuracy and loss through the 1500 iterations of training. Note that the graph is split into two figures due to Google Colab being interrupted after 1000 iterations, so we had to retrain the model for the remaining 500 iterations.

As seen in the above figures, our accuracy was unfortunately low. We believe this is likely due to the limited training data we provided to train our model. The complete Caltech Pedestrian Dataset [1] had six sets designated for training, but we only used one set for both training and testing. We made this compromise due to limitations we faced on Google Colab. In converting the entire dataset to the YOLO format, we resulted in over 100 gigabytes of images. Due to Colab's has limited storage and GPU runtime, we were unable to use all the images we would have hoped for to train our model.

Another issue we faced was that our model consistently failed to recognize pedestrians that were close to the camera. We believe that this is because most of the training data provided in our dataset contains pedestrians that are far from the camera. Thus, when pedestrians walk past the camera directly, the model will fail to recognize this. Going forward, we hope to improve our detection model by using a wider range of images from the training sets, as we believe that to be the main source of our issues.

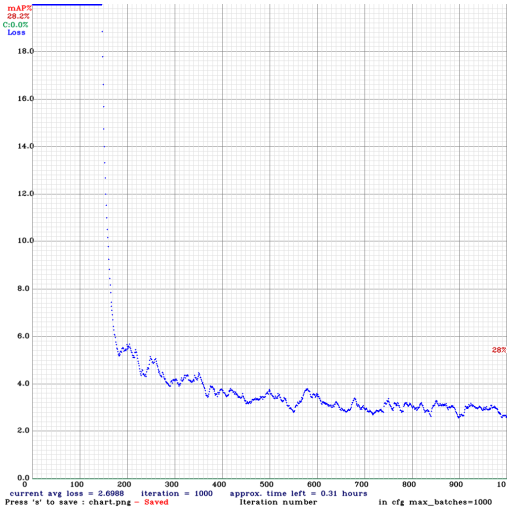

Figure 1. Loss and accuracy graph for YOLO model training over the first 1000 iterations

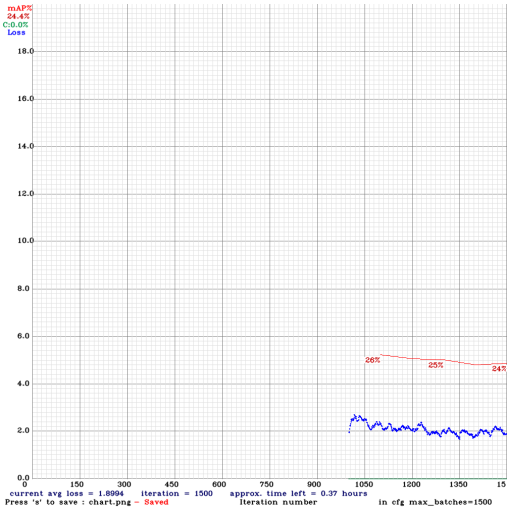

Figure 2. Loss and accuracy graph for YOLO model training from the 1000th to 1500th iterations

3.2. Tracking Evaluation

The quantitative and qualitative results of our tracker is presented in this section. The evaluating tool for multi-object tracking, py-motmetrics [4], is utilized to calculate the quantitative metrics whose meaning are listed in Table I. More information and the definition of the metrics can be found in [7].

To evaluate the performance of our multi-pedestrian tracker, TUD-Campus data in the popular dataset MOT Challenge is used. The tracking results are shown in

Metrics	Description	TUD-Campus Results
IDP	Global min-cost precision for ID measures	59.0%
IDR	Global min-cost recall for ID measures	58.2%
IDF1	Global min-cost F1 score for ID measures	58.6%
MOTA	Multiple object tracker accuracy	41.8%
MOTP	Multiple object tracker precision	76.5%
GT	Number of ground truth objects	8
MT	Number of objects tracked for at least 80 percent of lifespan	4
PT	Number of objects tracked between 20 and 80 percent of lifespan	4
ML	Number of objects tracked less than 20 percent of lifespan	0

Table 1. The Quantitative Results of the Tracker on TUD-Campus Data.

Figure 3(a) and the quantitative results are listed in Table 1. Among the benchmarks, Multiple Object Tracking Accuracy (MOTA) and Multiple Object Tracking Precision (MOTP) is the two widely used index for evaluation. Our tracker gets 41.8% and 76.5% for MOTA and MOTP respectively. Moreover, for track quality measures the ground truth (GT) trajectory can be classified as mostly tracked (MT), partially tracked (PT), and mostly lost (ML). In our test, our tracker gets 50% MT, 50% PT, and 0% ML. Though the performance is not perfect (i.e. the percentage in MT is not extremely high), the tracker almost does not lose tracked objects.

3.3. The Results on Dataset

Our YOLO detector and multi-pedestrian tracker are combined to test on the Caltech Pedestrian Dataset. Parts of the results are shown in Figure 3(b). Note that since the dataset does not provide the information of the pedestrian ID, the quantitative metrics cannot be evaluated. Our framework can detect and track the pedestrian while the car driving. However, for some cases, it fails because the results are base on the performance of both detector and tracker. For instance, the pedestrians crossing the road are not tracked since this case is rare in the training data so the detector fails to catch them. Also, when the speed of the car is fast, the tracker fails and recognizes the same pedestrian as different one. The complex cases in real world video need to be further considered to improve the performance of our work.

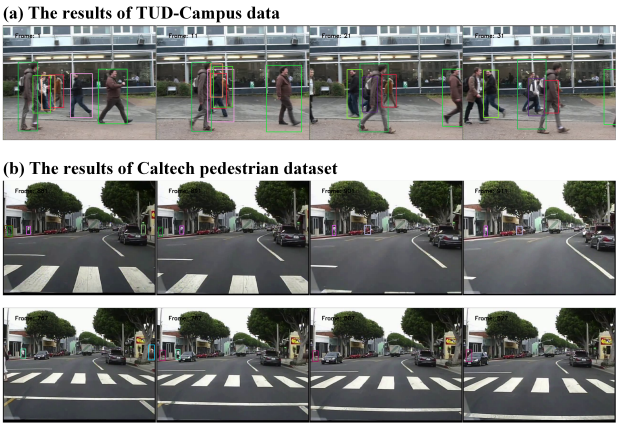

Figure 3. The tracking results of (a) TUD-Campus data and (b) Caltech Pedestrian Dataset

4. Implementation

The object identification portion, YOLO, was based on YOLO v4 pretrained weights from github.com/AlexeyAB and a custom data set. We trained the model ourselves. The tracking part of our project was built on the tracking library of OpenCV. We utilized their Multitracker to keep track of all the entities we were interested in. We built a system that could read in a video to be tracked and text file detailing the position of the entities given by YOLO and would automatically alternate between the tracker and YOLO. Using a combination of IoU, Intersection over Union, and coefficient of correlation we are able to more accurately locate and get the pathing of an object as well as determine whether or not it should be tracked.

References

[1] Caltech pedestrian detection benchmark. http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/, 2009. 2, 3

[2] C.-Y. L. H.-Y. M. Bochkovskiy, Alexey; Wang. Yolov4: Optimal speed and accuracy of object detection. *eprint arXiv:2004.10934*, 2020. 2

[3] C. Han, C. Wang, E. Mei, J. Redmon, S. Divvala, Z. Wu, X. Wang, Y.-G. Jiang, H. Ye, and X. Xue. Yolo-based adaptive window two-stream convolutional neural network for video classification. 2017. 1

[4] C. Heindl. py-motmetrics: Python implementation of metrics for benchmarking multiple object trackers. https://github.com/cheind/py-motmetrics, 2020. 3

[5] W. Lan, J. Dang, Y. Wang, and S. Wang. Pedestrian detection based on yolo network model. In *2018 IEEE International*

Conference on Mechatronics and Automation (ICMA), pages 1547–1551, 2018. 1

[6] A. Lukezic, T. Vojir, L. Cehovin Zajc, J. Matas, and M. Kristan. Discriminative correlation filter with channel and spatial reliability. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6309–6318, 2017. 2

[7] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016. 3

[8] J. Shi, X. Wang, and H. Xiao. Real-time pedestrian tracking and counting with tld. *Journal of Advanced Transportation*, 2018:1–7, 10 2018. 1

[9] S. Zachau. Caltech pedestrian dataset to yolo format converter. https://github.com/simonzachau/caltech-pedestrian-dataset-to-yolo-format-converter, 2018. 2