Improving Pedestrian Safety Analysis with Tracking

Andrew Magill
Texas State University
San Marcos, TX
andrewmagill@txstate.edu

Abstract

Researchers at the Texas Advanced Computing Center, in conjunction with the City of Austin, have created a system for analysing traffic safety [1] using multiple object tracking with video captured at city intersections. The authors state that tracking pedestrians is a challenge requiring improved methods to increase the accuracy and utility of the analysis system. In this paper, I attempt to improve tracking accuracy with minor modification to the original algorithm that do no require additional training. I also compare the performance of the original and modified algorithms to the current state of the art methods, and suggest future work.

1. Introduction

In a study of traffic safety, researchers at the Texas Advanced Computing Center and the City of Austin have described a method for tracking vehicular and pedestrian traffic at city intersections [1]. The method uses a CNN trained on seven classes (person, car, bus, truck, bicycle, motorcycle, and traffic light) to detect objects in each video frame. A distance measure is used to match detections between frames, and predict object locations in future frames, forming trajectories for each object.

This method does not perform equally well on each class of detected objects. Pedestrians prove challenging to track, often abruptly changing velocity or direction, and frequently occluding each other from the perspective of the camera. These challenges are most pronounced in crowded scenes with deep depth of view, as may be found in video captured by traffic cameras. Improved methods are needed to track these more complex targets.

I have evaluated the performance of the original algorithm with several modifications. The original algorithm uses intersection over union (IOU) to calculate similarity between object detections and existing tracks. Another measure of similarity may be used in place of IOU without requiring further changes to the algorithm. I have substituted the original measure with a Euclidean distance calculation, and methods for determining visual similarity scores, in an attempt to improve performance. I have also introduced memory of occluded tracks for possible re-identification, and finally I have adjusted the mechanism for predicting locations in future frames.

I provide a benchmark of these modifications applied to popular dataset provided by the annual MOTChallenge [7], a competition for evaluating state-of-the-art tracking algorithms, using the common tracking metrics, MOTA [8] and IDF1 [9].

The results show modest improvement over the original algorithm, however, a large gap remains between the results of these modifications and the current state-of-the-art algorithms for tracking pedestrians, Tracktor and Tracktor++ [3]. I suggest that Tracktor may substitute for the original method without additional training beyond that which is needed for detection. The code used for the modifications and evaluation are publicly available on GitHub[1].

2. Data

The City of Austin does not regularly record streaming video from overhead mounted traffic cameras, with exceptions only for research purposes, such as the traffic analysis study in question. Due to privacy concerns, the video from the study has not been made publicly available.

Fortunately, the Multiple Object Tracking Benchmark[2] provides many annotated pedestrian datasets. I have chosen a video sequence with a scene similar to that of an overhead mounted traffic camera. The sequence has a deep depth of view, a crowded pedestrian scene, elevated viewpoint, and steady camera. The video sequence, recorded at 30 frames per second, contains 43 pedestrians on average per frame over 1050 frames with resolution 1920x1080 pixels. A snapshot of each video can be seen for comparison in Figure 2.

The MOTChallenge video sequences are accompanied by training and test data, with pedestrian detections and

[1] https://github.com/andrewmagill/motaustin
[2] https://motchallenge.net/

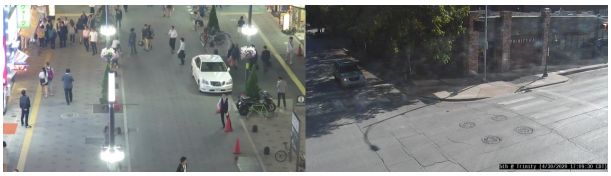

Figure 1. Left: The first frame of the MOT17-04 video sequence, Right: an intersection mounted camera operated by the City of Austin.

ground truth tracking annotations.

3. Methods

The method described by Xu et al. involves two components: a detector, and a tracker. The detector has been trained on several classes of objects, including cars, buses, and bicyclists [1], in this paper I will refer only to pedestrian detections. The detector identifies objects in each video frame and localizes each object by drawing around it a tightly enclosing bounding box.

With these detections, the location and class of all objects is known, but there is no known relationship between the detections in each frame. It is the job of the tracking component to measure relationships between objects detected in each frame and construct trajectories of objects over time.

The tracker in question uses location, calculated as the intersection over the union (IOU) of bounding boxes, to measure the similarity of objects between frames. This measure may be replaced, however, without requiring any additional changes to the algorithm. I have evaluated the performance using IOU, as originally described, Euclidean distance, visual similarity, and a combination of these methods.

3.1. Detector

The original algorithm, developed by researchers at TACC, and the City of Austin, uses the YOLOv2 detection algorithm with a Darknet-19 architecture [1]. YOLOv2 is a real-time object detector that initializes predetermined anchor boxes over an image and optimizes a classification loss function to find the center of detected objects, and a localization loss function to refine anchor box offsets to tightly bound detected objects [4].

Many advancements in object detection algorithms have been made since the publication of YOLOv2, in fact the fourth iteration of this algorithm, YOLOv4 [5], has been recently released. With more accurate algorithms in common use, I have decided not to benchmark the original algorithm with a YOLOv2 detector, and have instead conducted all benchmarks using Faster R-CNN with a ResNet-101 architecture and Pyramid Feature Network. I was able to do this without training, as the authors of the current state-of-the-art tracker, Tracktor++ [3], have made publicly available weights for an architecture trained on the MOT17 dataset[3].

In similar fashion to YOLOv2, Faster R-CNN also initializes bounding boxes with predetermined anchor boxes of several sizes and aspect ratios. The algorithm locates objects using a Region Proposal Network (RPN) [2], which takes feature maps from a CNN and produces class independent region proposals with an objectiveness measure, indicating whether the features contained in the boundary box represent the foreground or background of the image.

I expect adoption of a more advanced detection algorithm will result in higher precision, and possibly recall, over the YOLOv2 detector; however, detection is not the component likely to see the largest gains. My work focused on finding improvements to the tracking component, which I will describe next.

3.2. Tracker

The tracking component attempts to match detections across video frames to construct object trajectories over time. These detections are represented by bounding boxes tightly fitting detected objects. The tracking algorithm attempts to match objects from one frame to the next based on similarity between the objects. This similarity is measured as the intersection over union (IOU) of bounding boxes across frames. The process works as follows: A set of trajectories is initialized to the detections found in the first frame. Detections in the second frame are matched to trajectories from the first. With two points in each trajectory, the angle and distance can be calculated, which is used to predict the location of the object in the next frame. Any detection in the second frame that does not match a trajectory from the first becomes a newly initiated trajectory. For all remaining frames, detected objects are matched to trajectories, with unmatched detections initializing new trajectories [1] (see Algorithm 1).

[3] https://github.com/phil-berman/tracking_wo_bnw

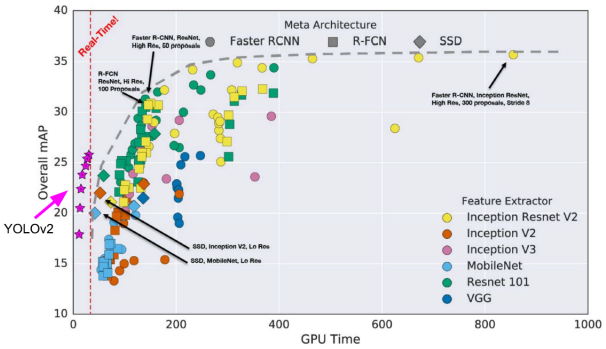

Figure 2. Benchmarks indicate Faster R-CNN would produce more precise detections than YOLOv2.

Algorithm 1: TRACKING

Input: N = { x_{ij} | i: frame index, j: object index}

Output: T= { t_{ij} | i: trajectory index, j: object index within this trajectory} as the set of objects stored by a list of trajectories

$T \leftarrow n_{ij} \in N, i = 0$

foreach $n_{ij} \in N, i > 0$ **do**

  foreach $n_{ij} \in N, i > 0$ **do**

    $dists \leftarrow dist(n_{ij}.location, pred(t_k, i))$

  if $min(dists) < threshold$ **then**

    $t_{argMin(min\_dists)} \leftarrow n_{ij}$

  else

    $T \leftarrow n_{ij}$

I have evaluated several modifications to this algorithm. The first replaces the calculation of IOU with a simple Euclidean distance calculation. Another modification adds a visual similarity measure. I have also included memory of deactivated tracks, so that trajectories for occluded objects may resume if the object is re-identified. Finally, I have modified the location prediction mechanism to accommodate the memory function.

3.3. Memory

The original algorithm predicts the next location determined by the distance and angle calculated from the previous observations. By including memory, and without any other modification to the algorithm, the location prediction mechanism would continue this motion in the same direction and at the same velocity throughout the memory window. I have evaluated two modifications: eliminating location prediction altogether for missing frames, the tracker searches for objects where they were last seen, and in an effort to split the difference I decrease the predicted motion by half for each frame during the memory window.

3.4. Spatial Similarity

The algorithm uses a distance threshold parameter to eliminate unlikely matches. In the original algorithm, this threshold applies to the ratio of intersection over union of bounding boxes, which has a minimum value of 0 and maximum 1. When calculating similarity by Euclidean distance, I return a distance score calculated as a ratio so that, similar to the IOU calculation, the distance score has a minimum value of 0 and maximum of 1 (see equation 1).

$$DistanceScore = \frac{SearchArea - Distance(box_a, box_b)}{SearchArea} \quad (1)$$

Increasing the threshold parameter results in fewer matched trajectories and more frequent newly initiated trajectories. Decreasing the threshold creates fewer new trajectories, but likely has the effect of causing many incorrect trajectory assignments. This helps to illustrate a problem with the memory window. For any significantly large window (1 second is 30 frames) you will need to increase the search area if you hope to successfully re-identify occluded objects. However, a larger search area may increase erroneous trajectory assignments. In an effort to reduce identity switching, I have evaluated the algorithm with two methods for determining visual similarity, and with Euclidean distance used to avoid an exhaustive search. Only objects within a radius of each other (across frames) are compared for visual similarity.

3.5. Visual Similarity

I conducted a preliminary analysis on several different visual similarity methods by extracting tracked objects across video frames with locations provided in the ground truth data. I use these clipped images to calculate the intra-track and inter-track (or across-track) similarity performance. I was interested in finding methods with high intra-track (true positive) scores and low inter-track (false positive) scores. Two algorithms performed very well, Scale Invariant Feature Transform (SIFT) [11], and Structural Similarity Index (SSIM) [12].

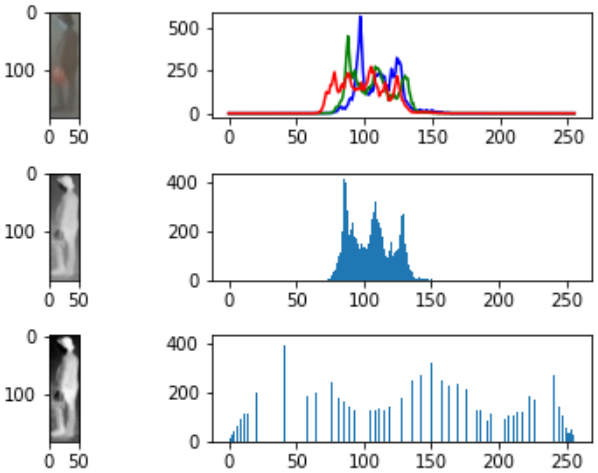

Figure 3. Example of a low contrast patch with histogram in RGB, grayscale, and grayscale after histogram equalization.

SIFT uses the derivative of a Gaussian filter to detect and describe keypoint features in the image. Keypoints are matched based on Euclidean distance between their vectors with a matching algorithm. SSIM looks for similarity between pixels, sort of the opposite to Mean Squared Error (MSE), and produces a similarity score and difference mapping.

The visual similarity methods performed less well on tracks that have low contrast between foreground and background. One such example can be seen in Figure 3 . SIFT was almost completely unable to detect features for this man. The histogram shows high intensity over a narrow range of color gradients. Histogram equalization is a process that attempts to more evenly distribute intensity over the full range of gradients. Slightly more formally, the probability that any given pixel will have any given intensity value should be close to equal for all pixels and intensities.

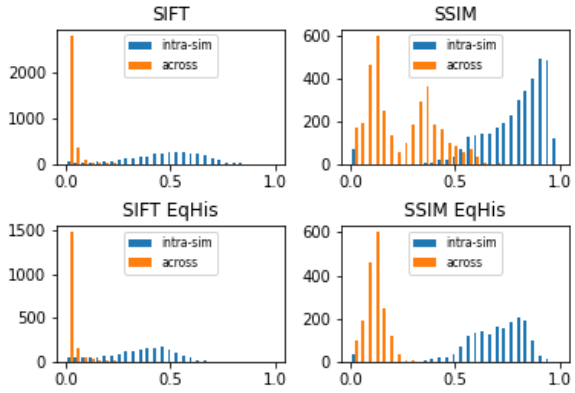

Figure 4. Performance of SIFT and SSIM on sample inter-track and intra-track similarity, without grayscale histogram equalization (top) and with equalization (bottom). A clear separation between intra-track and inter-track scores is desirable.

		Original		Equalized	
		Within	Across	Within	Across
SIFT	max	1.00	0.54	0.77	0.27
	min	0.00	0.00	0.00	0.00
	mean	0.48	0.03	0.37	0.02
	stdev	0.18	0.06	0.15	0.04
SSID	max	0.98	0.74	0.95	0.34
	min	0.00	0.00	0.00	0.00
	mean	0.80	0.24	0.72	0.11
	stdev	0.17	0.16	0.15	0.05

Table 1. Statistical performance of SIFT and SSID within and across tracks, and with and without grayscale histogram equalization on sample data. Higher intra-track similarity values, and lower inter-track similarity are desirable.

Histogram equalization decreased both true an false positives for both algorithms, however false positives were reduced further, particular using SSIM. A clear separation becomes evident in the distribution of true positive and false positives SSIM scores, as can be seen in Figure 4. With this information we can better pick a potentially good similarity threshold.

3.6. Evaluation

The MOTChallenge has produced a development kit[4] that evaluates MOT tracking hypotheses against ground truth to produce a broad set of metrics. This development kit runs in MATLAB, I have instead used a very nice Python implementation available on GitHub[5].

MOTA [8] and IDF1 [9] are the most commonly used metrics to evaluate tracking performance. MOTA a ratio of false negatives, false positives, and identity switching (all the bad stuff) over true positives (see equation 5). MOTA scores have an upper bound of 1 and lower bound of negative infinity.

IDF1 is the harmonic mean between Identity Precision (IDP) and Identity Recall (IDR). IDP is the ratio of true positives to true positive and false positives, IDR is the ratio of true positives and false negatives.

$$IDP = \frac{IDTP}{IDTP + IDFP} \tag{2}$$

$$IDR = \frac{IDTP}{IDTP + IDFN} \tag{3}$$

$$IDF_1 = \frac{2IDTP}{2IDTP + IDFP + IDFN} \tag{4}$$

$$MOTA = 1 - \frac{IDFN + IDFP + IDs}{IDTP} \in (-\infty, 1] \tag{5}$$

4. Results

The results show the best improvement over the original algorithm using Euclidean distance (L2 norm) alone as a similarity measure, with memory. The location prediction mechanism had no effect on the results with this measure, continuous motion, decreasing velocity, and no motion prediction, all produced the same results. I was disappointed to find that the visual similarity methods did not improve on the simple Euclidean distance results, and surprised to see that SIFT performed better than SSIM. You can see in Figure 4 that the choice of similarity threshold is perhaps less clear than originally indicated by preliminary analysis. Further analysis may suggest threshold values that produce better results, however, I would strongly suggest that the TACC and City of Austin researchers adopt the Tracktor method of tracking, as even the detector-only version of the algorithm greatly outperforms any other approach.

The results show modest improvement over the original algorithm, however, the current state-of-the-art algorithms for tracking pedestrians, Tracktor and Tracktor++ [3] show significant improvement over any modification to the original algorithm.

[4] https://motchallenge.net/devkit
[5] https://github.com/cheind/py-motmetrics

method	IDF1	IDP	IDR	IDs	MOTA
original*	55.8%	75.2%	44.3%	**14**	50.9%
IOU c	60.6%	81.5%	48.3%	51	51.2%
IOU d	61.2%	82.4%	48.7%	52	51.2%
IOU s	61.7%	82.9%	49.1%	39	**51.3%**
L2	**61.9%**	**83.2%**	**49.3%**	33	**51.3%**
SSIM	59.4%	79.9%	47.3%	115	51.1%
SIFT	**61.2%**	**82.2%**	**48.7%**	**43**	**51.3%**
Tracktor*	**71.3%**	**90.6%**	**58.8%**	22	**64.5%**
Tracktor++	71.2%	90.5%	58.7%	**21**	64.5%

Table 2. Results for all tracking approaches. *These algorithms do not implement re-identification.

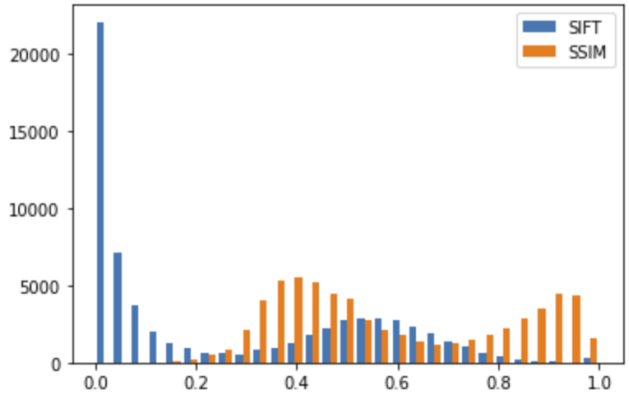

Figure 5. Histogram of SSIM and SIFT similarity comparison scores for over the entire video sequence (objects are only compared to other objects within a set 2d radius).

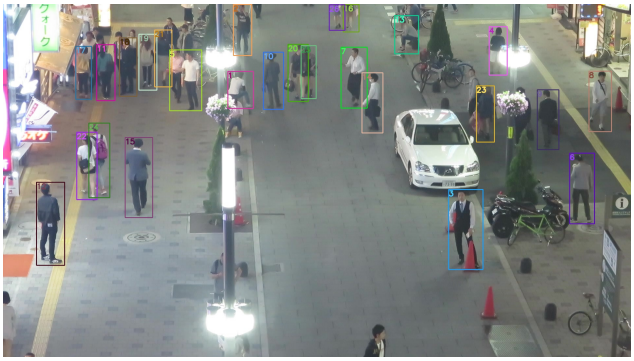

Figure 6. Tracked objects in the video frame.

5. Future Work

Although Tracktor outperforms my results by a wide margin, there is still room for improvement in tracking performance. I believe the main issue with common current approaches is that we are attempting to track three dimensional objects on a two dimensional plane. Parameters that work well at the bottom of the image (closest to the camera) are too large near the top (furthest from the camera), and vice versa. Visual similarity methods (including Siamese CNN [6]) deserve further attention in my future efforts. Other approaches may try to create 3D models of tracking targets, or otherwise consider a three dimensional space. Perhaps depth can be estimated at the top of the frame, with parameters adjusted for objects higher in the frame (deeper in the field of view). As a last thought, I would begin to improve on the pedestrian safety study by replacing the current detector and tracker approach used in the study with the detector only approach of Tracktor [3].

References

[1] Xu, W., Ruiz-Juri, N., Huang, R., Duthie, J.C., Clary, J.J. (2018). Automated pedestrian safety analysis using data from traffic monitoring cameras. SCC '18.

[2] Ren, S., He, K., Girshick, R.B., Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39, 1137-1149.

[3] Bergmann, P., Meinhardt, T., Leal-Taixe, L. (2019). Tracking Without Bells and Whistles. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 941-951.

[4] Redmon, J., Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 6517-6525.

[5] Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. ArXiv, abs/2004.10934.

[6] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Saeckinger, and Roopak Shah. (1993). Signature verification using a "siamese" time delay neural network. Conference on Neural Information Processing Systems.

[7] Milan, A., Leal-Taixé, L., Reid, I.D., Roth, S., Schindler, K. (2016). MOT16: A Benchmark for Multi-Object Tracking. ArXiv, abs/1603.00831.

[8] Bernardin, K., Stiefelhagen, R. (2008). Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. EURASIP Journal on Image and Video Processing, 2008, 1-10.

[9] Ristani, E., Solera, F., Zou, R.S., Cucchiara, R., Tomasi, C. (2016). Performance Measures and a Data Set for Multi-target, Multi-camera Tracking. ECCV Workshops.

[10] Henschel, R., Marcard, T.V., Rosenhahn, B. (2019). Simultaneous Identification and Tracking of Multiple People Using Video and IMUs. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 780-789.

[11] Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, 60, 91–110.

[12] Wang, Zhou; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. (2004-04-01). "Image quality assessment: from error visibility to structural similarity". IEEE Transactions on Image Processing. 13 (4): 600–612.