Time-First Tracking: An Efficient Multiple-Object Tracking Architecture for Dynamic Surveillance Environments

Joachim Lohn-Jaramillo[1], Khari-Elijah Jarrett[1], Laura Ray[1], Richard Granger[2] and Elijah Bowen[2]
[1]*Thayer School of Engineering, Dartmouth College, 14 Engineering Drive, Hanover, U.S.A.*
[2]*Department of Psychological and Brain Sciences, Dartmouth College, 3 Maynard St., Hanover, U.S.A.*

Keywords: Multiple Object Tracking, DBSCAN, Background Subtraction, and Tracking by Detection.

Abstract: Given the countless hours of video that are generated in surveillance environments, real-time for multi-object tracking (MOT) is vastly insufficient. Current MOT methods prioritize tracking accuracy in crowded environments, with little concern for total computational expense, which has led to a reliance on expensive object detectors to perform tracking. Indiscriminate use of object detectors is not scalable for surveillance problems and ignores the inherent spatio-temporal variation in scene complexity in many real-world environments. A novel MOT method is proposed, termed "Time-First Tracking", which relies on "shallowly" processed motion with a new tracking method, leaving the use of expensive object detection methods to an "as-needed" basis. The resulting vast reduction in pixels-processed may yield orders of magnitude in cost savings, making MOT more tractable. Time-First Tracking is adaptable to spatio-temporal changes in tracking difficulty; videos are divided into spatio-temporal sub-volumes, rated with different tracking difficulties, that are subsequently processed with different object localization methods. New MOT metrics are proposed to account for cost along with code to create a synthetic MOT dataset for motion-based tracking.

1 INTRODUCTION

Current MOT methods are primarily "frame-first" tracking methods that focus on the visual details of each frame, instead of focusing on the information that changes over time like motion. A focus on motion first is termed "time-first" tracking in this paper. The MOT Challenges serve as benchmark datasets in MOT (Dendorfer et al., 2019, 2020; Leal-Taixé et al., 2015; Milan et al., 2016). The datasets focus on short video clips with dense, largely homogenous target distributions. Tracking in these scenes is very hard and often done with Tracking-by-Detection (TbD) methods. TbD methods join sequential detections together to create tracks, where detections are provided from a trained object detector, e.g., Ren et al. (2015). Sometimes visual tracking tools or point tracking strategies are added to improve tracking output (Bochinski et al., 2019; Keuper et al., 2018) at the cost of lower processing speeds. The frame-first approach of object detection and motion estimation are computational bottlenecks that limit speeds to near real-time. Researchers treat real-time processing speeds as adequate, but London alone has 500,000 surveillance cameras yielding more than 10 million video hours per day; lacking dedicated processing per camera, real-time is vastly inadequate for both MOT methods and the higher-level tasks that could use tracking outputs as priors.

Precise object localization may be necessary in complex scenes, but long-duration videos are quite simple much of the time. For example, a camera at a train station will not see the same crowd density at rush hour as it will at midnight, nor will it see the same number of people on the tracks versus the platforms. In uncongested scenes, imprecise motion estimation, such as a simple background subtraction routine (Zivkovic, 2004) can localize objects in place of detectors at a fraction of the cost. Typically, TbD methods struggle to use these detections because they are inconsistent and do not always correspond to single objects. The MOT Challenge datasets overlook these applications of TbD methods by focusing on "hard" videos; "easy" videos and their background models are not included for comparison. One may reason that if a tracking method can manage a hard scene, then it can manage an easy one. This is only true when cost and time constraints are ignored,

Lohn-Jaramillo, J., Jarrett, K., Ray, L., Granger, R. and Bowen, E.
Time-First Tracking: An Efficient Multiple-Object Tracking Architecture for Dynamic Surveillance Environments.
DOI: 10.5220/0010304906020611
In *Proceedings of the 10th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2021)*, pages 602-611
ISBN: 978-989-758-486-2

otherwise, researchers may consider more efficient solutions for easy scenes. It would be beneficial for a MOT method to adapt its approach to variations in spatio-temporal tracking complexity, so that the cost of the current approach is proportional to the current difficulty level. We introduce a proof-of-concept Time-First Tracking (TFT) architecture that divides a video into spatio-temporal sub-volumes, classified with different target densities, that are then processed with different object localization methods for tracking. TFT addresses two deficiencies in current MOT methods, which are that 1) current methods are not adaptable to imprecise motion estimation, which imposes a ceiling on processing speed and 2) current methods treat tracking difficulty as a spatio-temporal-independent variable, which is unrealistic. Other contributions include the formation of a new tracklet generation method, newly proposed cost metrics for MOT, and code for developing a new synthetic motion-based MOT dataset.

2 RELATED WORK

Tracking methods require object localization methods to measure object states. Object localization can be either detection-based or motion-based. Object detection is the identification of a specific type of object with a trained system, e.g., Felzenszwalb et al. (2010), Redmon et al. (2016), Ren et al. (2015), or Sadeghi & Forsyth (2014). These methods achieve good performance on benchmarks like PASCAL VOC (Everingham et al., 2014), but are expensive and rely on dedicated hardware to reach real-time speeds (Canziani et al., 2017). Alternatively, motion estimation methods can localize moving objects by finding where in the frame pixel values are changing; motion estimation is not object specific and varies in precision, from the pixel to the region level.

2.1 Motion Estimation

Here, "imprecise motion estimation methods," are the class of methods that use statistical background models and/or frame differencing methods to first detect foreground areas in a frame and then separately track detections over time to infer regional motion. These approaches are computationally cheap, but often inaccurate in unconstrained environments where there are crowds or illumination changes. Gaussian Mixture Models (GMMs) (Wren et al., 1997) are used to generate a statistical model for each pixel's intensity values; uncommon values are foreground, while common values are background. GMMs can be adjusted over the course of a video (Stauffer & Grimson, 1999; Zivkovic, 2004). In general, the quality and complexity of motion estimation in stationary cameras range from dynamic GMMs to the simplest frame-difference methods, e.g., Migliore et al. (2006), Singla (2014), and Zhan et al. (2007). Sobral & Vacavant (2014) and W. Kim & Jung (2017) overview the various methods and relevant datasets that exist in the field today.

"Precise motion estimation methods" explicitly estimate motion at the pixel level instead of the region level. Optical flow methods and their analogues (Farnebäck, 2003; Horn & Schunck, 1981; Lucas & Kanade, 1981) are primary examples, which analyze local changes in brightness patterns from frame-to-frame and create vector fields to describe the motion of each pixel. Optical flow methods remain a front end for a large body of computer vision tasks. However, due to their complexity and high attention to detail, they are generally slower than real-time (Simonyan & Zisserman, 2014; Zhang et al., 2016).

2.2 Tracking Methods

MOT methods are designed to maintain/update multiple tracks and correctly assign new detections to the appropriate tracks. The MOT Challenges and UA-DETRAC (Wen et al., 2020) are benchmarks for tracking performance, where trackers are evaluated according to the now common tracking metrics from Bernardin & Stiefelhagen (2008) and Li et al. (2009). These benchmarks have encouraged significant progress in TbD techniques, which share many concepts with data association, a mature field of research. Works such as Multiple Hypothesis Tracking (MHT) (Reid, 1979) and Joint Probabilistic Data Association (JPDA) (Fortmann et al., 1983) are early examples of systematic mathematical solutions to tracking that are still applicable, but slow. Recent efforts have improved the efficiency of MHT and JPDA (C. Kim et al., 2015; Rezatofighi et al., 2015), but they are still slower than real-time (Bewley et al., 2016). To increase speeds, some methods have been adapted to link small groups of detections, "tracklets," rather than individual detections, e.g., Benfold & Reid (2011), Dicle et al. (2013), Huang et al. (2008), and Perera et al. (2006).

MOT can also be made into a graphical problem, where detections are nodes and interactions between detections are edges (Bansal et al., 2004; Tang et al., 2015; Wen et al., 2014). These methods calculate trajectories by balancing the cost between local pairs of detections and the total cost of the graph. "Network flow" methods, e.g., Berclaz et al. (2011) and

Pirsiavash et al. (2011), are related to graphical approaches, but also include “sources” and “sinks,” which direct trajectory paths for optimization.

The vast majority of leading MOT methods in the MOT Challenges run below 30 frames per second (fps) with only a few exceptions. Simple Online and Realtime Tracking (SORT) (Bewley et al., 2016) shows faster than real-time on the MOT16 dataset and relies on a Kalman Filter (KF) (Kalman, 1960) for motion estimation and the Hungarian Algorithm (Munkres, 1957) for detection assignment. The joint KF-Hungarian (KF-H) approach is a common and efficient method, included in MATLAB 2020b (2020a). Another exception is the IOU tracker (Bochinski et al., 2017), which builds tracks based solely on the overlap of detections (intersection-over-union) between frames. This approach is extremely fast, running over 1,000 fps, but relies on consistent detections; minor temporal gaps in detections can cause identity switches. Bochinski et al. (2019) added a visual tracker to the IOU method, termed the V-IOU tracker, which improved accuracy but was an order of magnitude slower. Current works can be found in the recent MOT Challenges and UA-DETRAC results.

3 METHODS

Leveraging motion and other visual cues beyond bounding box proposals is an intuitive approach to tracking, but when used in conjunction with object detectors, gains in accuracy come at a high computational cost. Our TFT architecture aims to increase tracking speeds by localizing objects with imprecise motion estimation methods rather than object detection methods where possible. Motion detections and object detections are tracked with our Accumulated Motion Approximation (AMA) tracker.

3.1 Motion Estimation

First, the resolution is reduced to make the input images smaller. The specific size factor we use is relative to the size of the smallest object we wish to detect. If the smallest object is (S x S) pixels, then we have found that the resolution can be reduced by a factor of roughly S/3 so that a (3 x 3) morphological filter can dilate results back to the original size. This step greatly reduces false foreground detections but can merge nearby targets into a single foreground region. In practice, we have not reduced the resolution much beyond a factor of five. Next, if a background image is available to serve as the background model, we subtract the current frame from the background and threshold the result to make a black-and-white, or “binary,” frame. Otherwise, background modeling is performed with an adaptive GMM (Stauffer & Grimson, 1999) implemented in MATLAB 2020b (2020b), where typically, default parameters are used with some minor hand-tuning. The binary frame is processed with three simple morphological steps to temporally smooth the results and reject noise. The current binary frame is filtered with a (3 x 3) majority filter. Then we find the union of the current binary frame with the prior two binary frames. Lastly, another majority filter is used on the result of that union to produce the final binary frame. Foreground detections from this frame are filtered by size and aspect ratio thresholds, which are based on the size of the objects we expect to track.

Each resulting foreground region is saved with a feature vector representation of $\varphi = [x, y, f, bX, bY, w, h, a]$, which encodes the location and size of the bounding box that surrounds the detection: (x, y) is the center of the box, f is the current frame, (bX, bY) is the top left corner of the box, (w, h, a) is the width, height, and area of the box. Our approach assumes that with long-duration video from a particular camera angle, scene parameters like expected target size are known, camera motion is minimal, and robust background models can be built over time.

3.2 AMA Tracklets

We introduce an approach to generate tracklets in noisy background subtraction environments. Shown in Figure 1 (a), the inputs are the center locations of the detected foreground regions from a sequence of τ frames, referred to as a “Bin” of frames. AMA uses DBSCAN (Ester et al., 1996) to cluster foreground-region locations for a Bin, shown in Figure 1 (b). Efficient indexing methods in DBSCAN avoid the

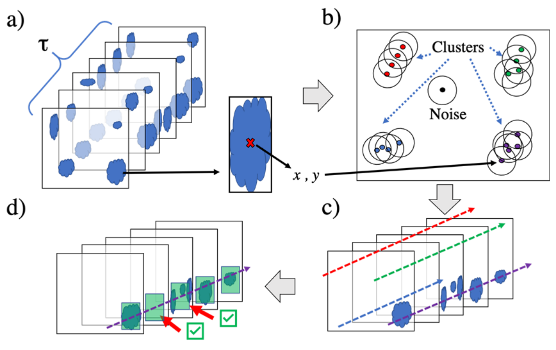

Figure 1: Generating AMA Tracklets. (a) Detected foreground regions in a Bin of τ frames. (b) DBSCAN clustering performed on detection locations (x, y). (c) Lines of best fit are calculated for each cluster. (d) Missing detections and partial detections are filled in.

calculation of distances between all points (Schubert et al., 2017). After clusters have been formed in the x-y plane, the frame-dimension is re-included to calculate three-dimensional orthogonal lines-of-best-fit, which are the estimated motion vectors for the clusters (Figure 1 (c)). AMA tracklets have the advantage of assigning multiple detections on one frame to the same cluster and filling in missing detections, which is shown in Figure 1 (d). This feature is useful in noisy background subtraction results, where there is not always a one-to-one match between objects and foreground regions.

Eps, Minpts, and τ are the primary parameters used to generate tracklets. While there are systematic methods to finding an optimal value for *eps* (Satopää et al., 2011), with foreground detections we choose to set this parameter based on the half-width of the average object we expect to track. This imposes a tracking speed limitation of (*eps* pixels/frame). *Minpts* is two because two detections are needed to infer a motion vector. Empirically, we have found that keeping τ small, ~10 frames, solves issues with inconsistent detections and limits the duration of any clustering errors, which can be fixed by future steps.

3.3 AMA Tracker

A set of AMA tracklets are found for each Bin of frames. Sequential Bins are overlapped so that they

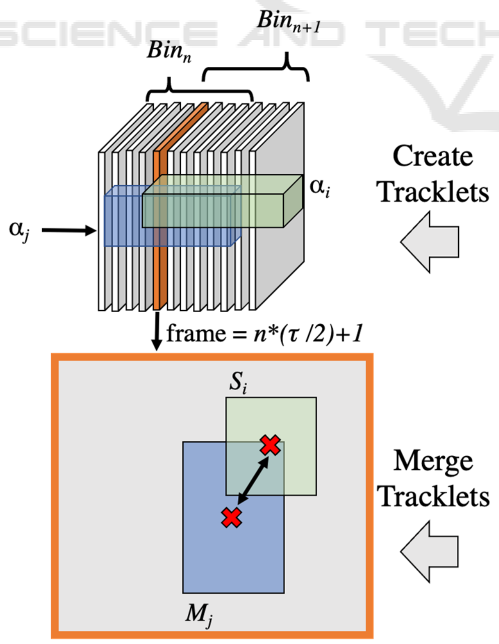

Figure 2: Merging AMA Tracklets. Tracklets are created for each Bin. At frame $n*(\tau/2)+1$, the bounding box from the middle of tracklet α_j in Bin_n (M_j) is compared with the starting bounding box from tracklet α_i in Bin_{n+1} (S_i).

share frames. For example, in Figure 2, Bin_n contains detections from frames 1-8, while Bin_{n+1} contains detections from frames 5-12. This was done because with background subtraction, foreground detections are inconsistent and can shift dramatically from one frame to the next. By sharing detections, sequential tracklets are easier to merge. To merge tracklets, the midpoints of each tracklet (M_j) in Bin_n are compared to the start points of each tracklet (S_i) in Bin_{n+1}. These points are bounding boxes, which are compared by 1) verifying that ratio of intersection area to union of area (IOU) between bounding boxes is above a threshold and 2) making sure the centers of the bounding boxes are within a threshold distance. These thresholds are hand-tuned for background subtraction but can be selected with training when using object detections from a dataset. Only tracklets from sequential Bins are merged, which can create identity switches when merging fails. This is a known deficiency that we plan to address in the future.

3.4 Time-First Tracking

The TFT method is shown in Figure 3 and the details of each step are described in the following sections. All steps are repeated for each sequence of frames, or "Stack." Unlike Bins, Stacks do not overlap. TFT intermittently measures target density and then finds targets with an object detector in high-density regions and with background subtraction in low-density regions. This reduces the usage of the object detector, which should increase overall processing speed but lower tracking accuracy in low-density regions.

3.4.1 Step 1: Stack Duration

Currently, the video is broken up into pre-defined Stacks, with a fixed depth of λ frames each, where λ is 60 frames. In the future, we plan to make λ dynamic so that the Stack depth can change as the speed and spatial distribution of targets changes. The first frame of each Stack is analyzed by a trained object detector, resulting in a set of proposed bounding boxes for the entire frame.

3.4.2 Steps 2 & 3: Sub-volume Creation

Detections from the first frame of a Stack are used to determine the spatial boundary between high- and low-density regions. We input target locations to DBSCAN to differentiate between high-density areas (clusters) and low-density areas (noise): *eps* is hand-tuned and *minpts* = 3. In high-target-density areas, it is hard to accurately localize objects. Clustered points

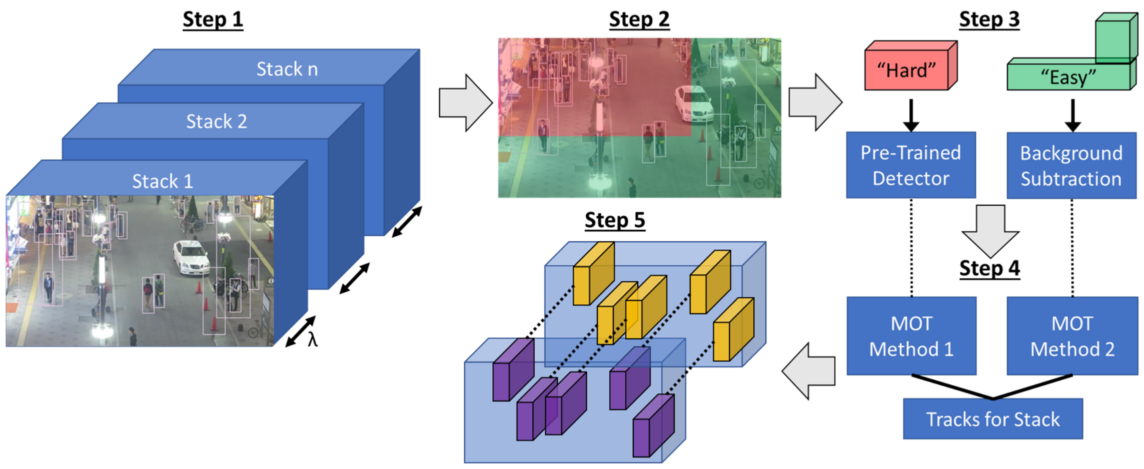

Figure 3: Current TFT Architecture. (Step 1) Split the video into Stacks, each containing λ frames. Run an object detector on the first frame of the Stack. (Step 2) Analyze object-detection density for the first frame of a Stack and then split the frame into two regions (high/low density). (Step 3) Keep spatial boundaries constant for all frames in the Stack to create sub-volumes. Pass pixels from the hard sub-volume to an object detector and pixels from the easy sub-volume to a background subtraction routine. (Step 4) Utilize separate trackers for the separate sub-volumes and merge the results. (Step 5) After each Stack is analyzed, connect tracks from adjacent Stacks to create contiguous tracks.

are given a "hard" label, while noise points are given an "easy" label. A naïve Bayes classifier is fit to the labeled locations. The classifier predicts labels for a grid of test points to form a single bounded hard region. The procedure for region segmentation is an area for future work.

Spatial boundaries are currently kept constant for each frame of a Stack, creating two sub-volumes with different target density levels. In the hard sub-volume, a trained object detector, e.g. Faster R-CNN (Ren et al., 2015), is used to localize objects for tracking. In the easy sub-volume, the background subtraction method from section 3.1 is used to detect moving objects. In the future we plan to expand TFT to handle multiple hard sub-volumes extruded in any direction, instead of being held in place.

3.4.3 Step 4: Tracking in Sub-volumes

The TFT architecture is setup to use two separate tracking methods for different sub-volumes, which allows current TbD methods to be easily adapted. In testing, we have implemented different combinations of the AMA tracker, a MATLAB (2020a) KF-H tracker, and the IOU tracker (Bochinski et al., 2017). Easy sub-volumes are the target of the AMA tracker, since noisy, imprecise results are expected here, but the AMA tracker can operate in both sub-volumes, with different parameters. When AMA uses object detections as inputs, *eps* and the IOU merge threshold are selected through training instead of being based on object size. Tracks are computed independently in each region and can begin in one region and terminate in another. Such tracks are merged with proximity and directional constraints to create a single set of tracks for a Stack.

3.4.4 Step 5: Connecting Stacks

To connect tracks between sequential Stacks, end points from tracks in Stack_n are compared to start points of tracks in Stack_{n+1} using IOU overlap and centroid distance thresholds. This is an expedient way to connect Stacks, but it is fragile to inter-frame inconsistencies. This is an area that will be further explored in future work.

3.5 Evaluation Criteria

In addition to the common metrics from Bernardin & Stiefelhagen (2008) and Li et al. (2009), we propose estimating detector-processing time by assuming a fixed speed, rather using a timed value, to avoid introducing hardware specific results. We find this reasonable, considering a range of detectors run between 1-200 fps, e.g., Cao et al. (2019), Redmon et al. (2016), Ren et al. (2015), and Sadeghi & Forsyth (2014). We estimate detector time (DT) for a particular video as

$$DT\ (s) = (nF * PixF) / (DS) \qquad (1)$$

where *nF* is the total number of frames in the video, *PixF* is the fraction of pixels of the video that were processed by the object detector, and *DS* is the assumed detector speed in frames per second. If detections from every full frame of the video are used, *PixF* = 1. Scaling detector time by *PixF* may not be exact in practice, due to overhead and architectural constraints, but it is a simple gauge of detector usage.

4 RESULTS

Preliminary results for the AMA tracker include an average MOTA score of 40.3 on the MOT17 training dataset with *eps* = 20, *minpts* = 2, IOU merge threshold of 0.65, and a distance threshold of *2*eps*. The average processing speed on a computer with a 2.4 GHz Intel i9 CPU is roughly 3,000 fps, excluding the time for object detections. The IOU tracker achieves an average MOTA of 47.4 at 4,000 fps, and a MATLAB KF-H tracker achieves an average MOTA of 46.9 at 900 fps.

These speeds are two orders of magnitude faster Than most published methods on the MOT Challenge and are only possible because visual features beyond object detections are ignored. This points to a need to decouple appearance-based tracking methods from motion-based tracking methods. A tracking method that ignores visual features would view a video frame like a background subtracted frame, where the detections are the only inputs. This is visualized in the top row of Figure 4, where a frame from a MOT Challenge video is beside a blank frame filled with the same object detections. Background subtraction can produce similar but less consistent bounding boxes, as shown in the second row of Figure 4 with a frame from the VIRAT dataset (Oh et al., 2011). Binary videos like this can be programmatically constructed, greatly expanding the number of labeled scenes for researchers to use. We have constructed a

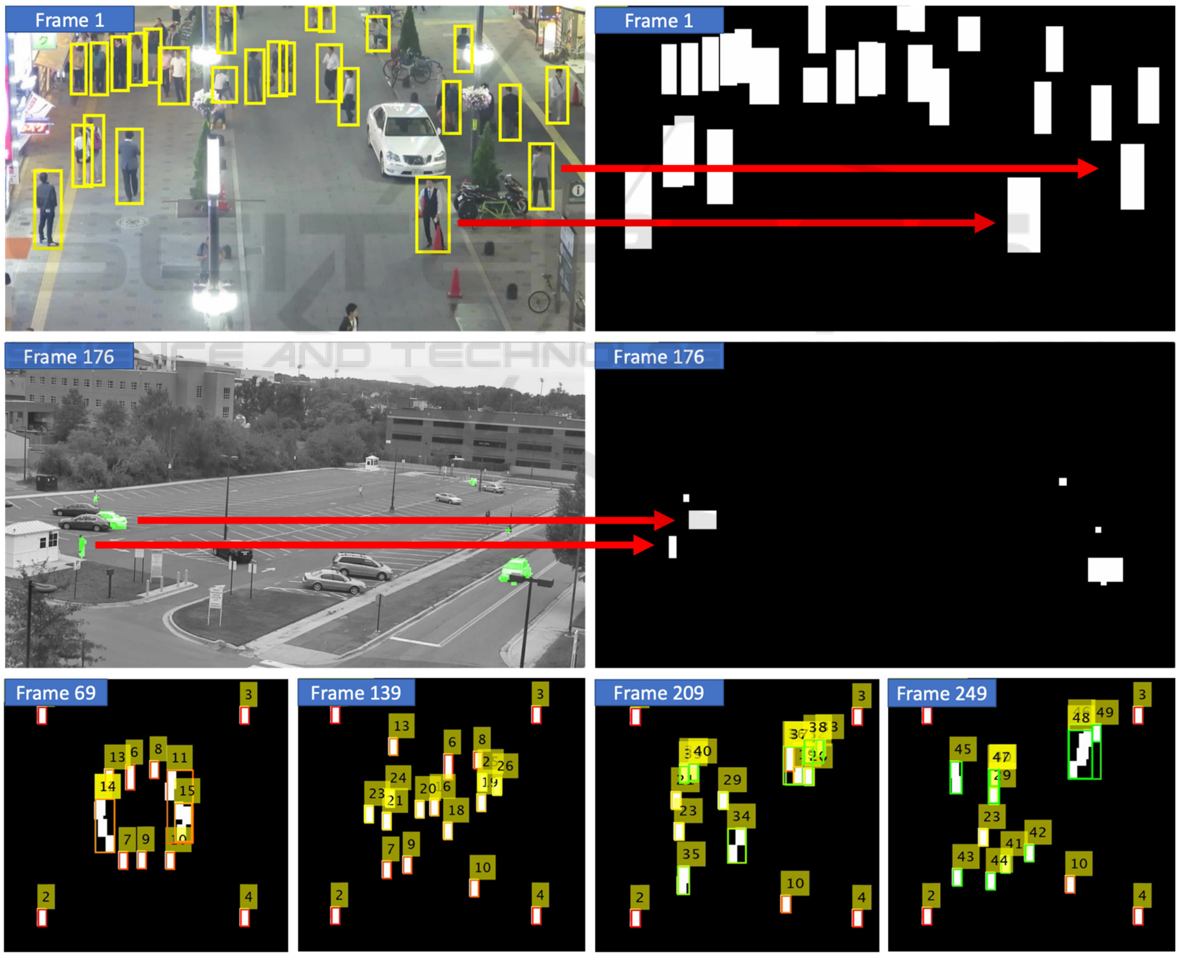

Figure 4: Ignoring Visual Features. In the top row, object detections are shown from a MOT Challenge video; detections are shown in yellow on the left and as white boxes on the right. In the middle row, foreground detections via background subtraction are shown from a VIRAT video; detections are shown in green on the left and as white boxes on the right. In the bottom row, a frame sequence is shown from our synthetic video, with the AMA tracker results overlaid.

tool to create these synthetic tracking videos, which model overhead, stationary cameras. Frames from an example video are shown in the bottom row of Figure 4 with the AMA tracker results overlaid. We plan to create a dataset where the number of targets can be varied, noise can be added, and false positives can be simulated. We are interested in comparing tracking methods across cross-sections of the dataset, e.g., measuring how processing speed varies with the number of targets.

TFT is designed for long duration videos from a stationary camera that exhibit variable target densities. Most MOT-oriented datasets are composed of short clips, do not contain background models, and primarily model high-target-density scenes, making the full datasets poor benchmarks for this approach. Qualitative results on one of the few overhead videos from the MOT Challenge are shown in the left column of Figure 5. The hard regions are shown in red and change location throughout the video. In the right column of Figure 5, another example is shown from a video in the VIRAT dataset using Aggregate Channel Feature detections (Dollar et al., 2014). The *PixF* value for the MOT Challenge video and the VIRAT video is 0.36 and 0.19, respectively. If a detector runs at 30 fps, the estimated detector time would be 12.6 and 4.7 seconds, respectively. Were the detector run on every full frame of the video (as usual), the estimated detector-processing time would increase to 35 and 25 seconds, respectively. The increase in speed of TFT comes with the introduction of visible tracking errors, e.g., track 179 in the left column of Figure 5. The aim of our future work is quantifying the trade-off between accuracy and cost, making the TFT architecture more dynamic, and processing more real-world videos.

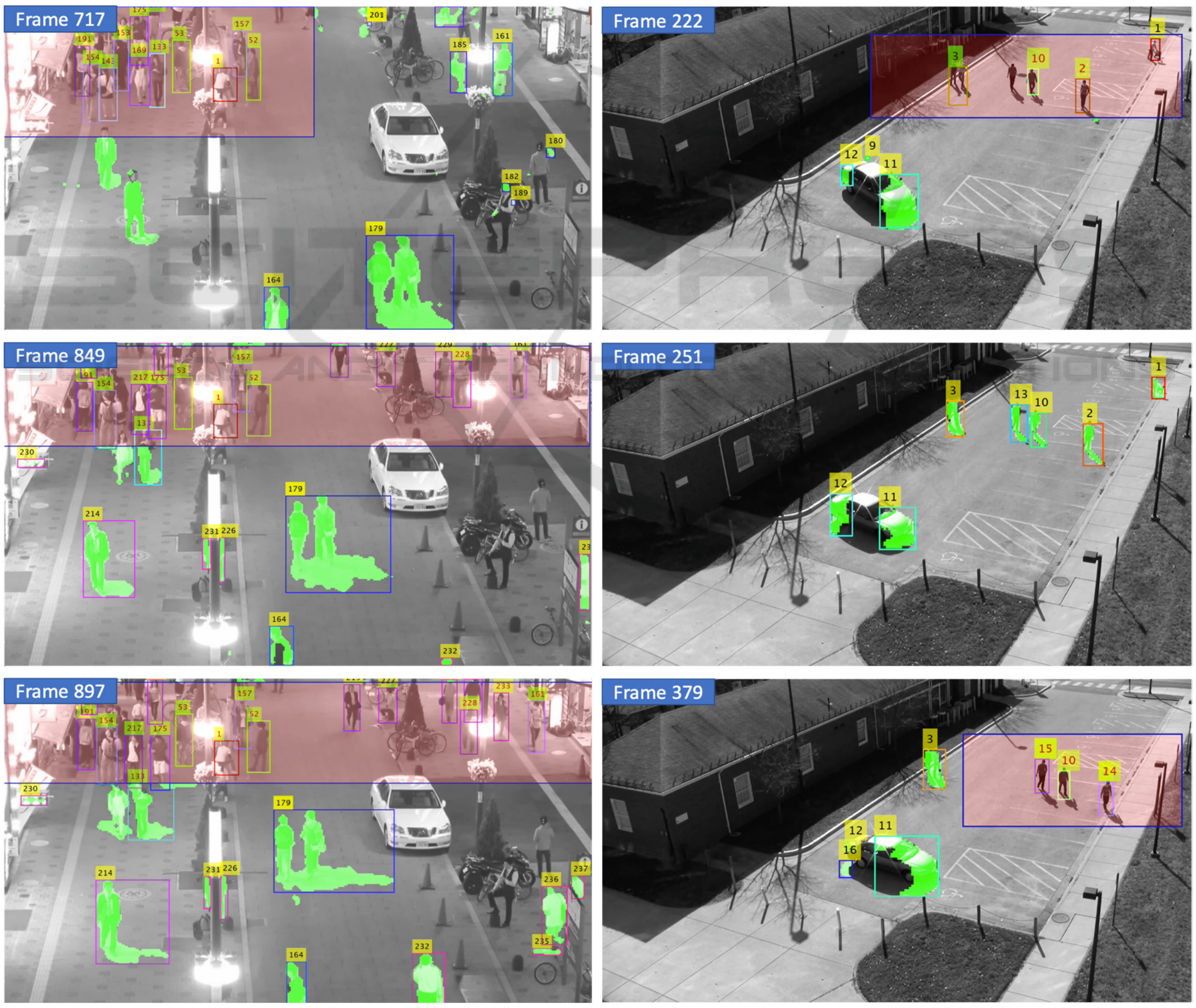

Figure 5: Qualitative Results for TFT. The frames in the left column come from a MOT Challenge video and the frames on the right column come from a VIRAT video. The hard region is the red shaded box, while the easy region is outside the box with foreground detections shown in green. Tracking errors are introduced in easy region, e.g., track 179 in left column.

5 CONCLUSIONS

In long-duration videos, hard and easy scenes are not separate problems and spatial target distribution can be quite non-homogenous. Access to dedicated hardware may be limited in surveillance settings, so MOT methods must prioritize efficiency and avoid the indiscriminate use of frame-first methods. The TFT architecture relies on cheap background modeling to handle scenes with simple tracking complexity, while reserving frame-first methods to an as-needed basis. We also show an adaptation of DBSCAN for MOT in our AMA tracker. Current work is aimed at building a new dataset for motion-based tracking methods, providing an extensive quantitative evaluation of our methods, and expanding the AMA and TFT methods to improve tracking errors after the first-pass track proposals have been made, e.g., Jarrett et al. (2019).

ACKNOWLEDGEMENTS

This research was supported in part by grant N00014-16-1-2359 from the Office of Naval Research.

REFERENCES

Bansal, N., Blum, A., & Chawla, S. (2004). Correlation Clustering. *Machine Learning*.

Benfold, B., & Reid, I. (2011). Stable multi-target tracking in real-time surveillance video. *IEEE Conference on Computer Vision and Pattern Recognition*, 3457–3464.

Berclaz, J., Fleuret, F., Türetken, E., & Fua, P. (2011). Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(9), 1806–1819.

Bernardin, K., & Stiefelhagen, R. (2008). Evaluating multiple object tracking performance: The CLEAR MOT metrics. *EURASIP Journal on Image and Video Processing*.

Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple online and realtime tracking. *2016 IEEE International Conference on Image Processing (ICIP)*, 3464–3468.

Bochinski, E., Eiselein, V., & Sikora, T. (2017). High-Speed tracking-by-detection without using image information. *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 1–6.

Bochinski, E., Senst, T., & Sikora, T. (2019). Extending IOU Based Multi-Object Tracking by Visual Information. *IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, 1–6.

Canziani, A., Culurciello, E., & Paszke, A. (2017). Evaluation of neural network architectures for embedded systems. *Proceedings - IEEE International Symposium on Circuits and Systems*.

Cao, Z., Hidalgo Martinez, G., Simon, T., Wei, S.-E., & Sheikh, Y. A. (2019). OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I., Roth, S., Schindler, K., & Leal-Taixe, L. (2019). CVPR19 Tracking and Detection Challenge: How crowded can it get? *Unpublished*.

Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I., Roth, S., Schindler, K., & Leal-Taixé, L. (2020). MOT20: a benchmark for multi object tracking in crowded scenes. *Unpublished*.

Dicle, C., Camps, O. I., & Sznaier, M. (2013). The way they move: Tracking multiple targets with similar appearance. *Proceedings of the IEEE International Conference on Computer Vision*.

Dollar, P., Appel, R., Belongie, S., & Perona, P. (2014). Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, 226–231.

Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2014). The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision*.

Farnebäck, G. (2003). Two-frame motion estimation based on polynomial expansion. *Proceedings of the 13th Scandinavian Conference on Image Analysis*, 363–370.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(9), 1627–1645.

Fortmann, T. E., Bar-Shalom, Y., & Scheffe, M. (1983). Sonar Tracking of Multiple Targets Using Joint Probabilistic Data Association. *IEEE Journal of Oceanic Engineering*.

Horn, B. K., & Schunck, B. G. (1981). Determining Optical Flow. *Artifical Intelligence*, *17*(1–3), 185–203.

Huang, C., Wu, B., & Nevatia, R. (2008). Robust Object Tracking by Hierarchical Association of Detection Responses. *European Conference on Computer Vision*, 788–801.

Jarrett, K., Lohn-Jaramillo, J., Bowen, E., Ray, L., & Granger, R. (2019). Feedforward and feedback processing of spatiotemporal tubes for efficient object localization. *International Conference on Pattern Recognition Applications and Methods*.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering, Transactions of the ASME*.

Keuper, M., Tang, S., Andres, B., Brox, T., & Schiele, B. (2018). Motion Segmentation & Multiple Object Tracking by Correlation Co-Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Kim, C., Li, F., Ciptadi, A., & Rehg, J. M. (2015). Multiple hypothesis tracking revisited. *Proceedings of the IEEE International Conference on Computer Vision*.

Kim, W., & Jung, C. (2017). Illumination-Invariant Background Subtraction: Comparative Review, Models, and Prospects. In *IEEE Access*.

Leal-Taixé, L., Milan, A., Reid, I., Roth, S., & Schindler, K. (2015). MOT challenge 2015: towards a benchmark for multi-target tracking. *Unpublished*.

Li, Y., Huang, C., & Nevatia, R. (2009). Learning to associate: Hybridboosted multi-target tracker for crowded scene. *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2953–2960.

Lucas, B. D., & Kanade, T. (1981). An Iterative Image Registration Technique with an Application to Stereo Vision. *Proceedings from the 7th IJCAI*, 674–679.

MATLAB 2020b. (2020a). *Motion-Based Multiple Object Tracking*. Mathworks Inc. https://www.mathworks.com/help/vision/ug/motion-based-multiple-object-tracking.html

MATLAB 2020b. (2020b). *vision.ForegroundDetector*. Mathworks Inc. https://www.mathworks.com/help/vision/ref/vision.foregrounddetector-system-object.html

Migliore, D. A., Matteucci, M., & Naccari, M. (2006). A revaluation of frame difference in fast and robust motion detection. *Proceedings of the ACM International Multimedia Conference and Exhibition*.

Milan, A., Leal-Taixé, L., Reid, I., Roth, S., & Schindler, K. (2016). MOT16: a benchmark for multi-object tracking. *Unpublished*.

Munkres, J. (1957). Algorithms for the Assignment and Transportation Problems. *Journal of the Society for Industrial and Applied Mathematics*, *5*(1), 32–38.

Oh, S., Hoogs, A., Perera, A., Cuntoor, N., Chen, C. C., Lee, J. T., Mukherjee, S., Aggarwal, J. K., Lee, H., Davis, L., Swears, E., Wang, X., Ji, Q., Reddy, K., Shah, M., Vondrick, C., Pirsiavash, H., Ramanan, D., Yuen, J., … Desai, M. (2011). A large-scale benchmark dataset for event recognition in surveillance video. *IEEE Conference on Computer Vision and Pattern Recognition*, 3153–3160.

Perera, A. G. A., Srinivas, C., Hoogs, A., Brooksby, G., & Hu, W. (2006). Multi-object tracking through simultaneous long occlusions and split-merge conditions. *IEEE Conference on Computer Vision and Pattern Recognition*, 666–673.

Pirsiavash, H., Ramanan, D., & Fowlkes, C. C. (2011). Globally-optimal greedy algorithms for tracking a variable number of objects. *IEEE Conference on Computer Vision and Pattern Recognition*, 1201–1208.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788.

Reid, D. B. (1979). An Algorithm for Tracking Multiple Targets. *IEEE Transactions on Automatic Control*.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems (NIPS)*.

Rezatofighi, S. H., Milan, A., Zhang, Z., Shi, Q., Dick, A., & Reid, I. (2015). Joint probabilistic data association revisited. *Proceedings of the IEEE International Conference on Computer Vision*.

Sadeghi, M. A., & Forsyth, D. (2014). 30Hz object detection with DPM V5. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.

Satopää, V., Albrecht, J., Irwin, D., & Raghavan, B. (2011). Finding a “kneedle” in a haystack: Detecting knee points in system behavior. *Proceedings - International Conference on Distributed Computing Systems*.

Schubert, E., Sander, J., Ester, M., Kriegel, H. P., & Xu, X. (2017). DBSCAN revisited, revisited: Why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems*.

Simonyan, K., & Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. *Advances in Neural Information Processing Systems*.

Singla, N. (2014). Motion Detection Based on Frame Difference Method. *International Journal of Information & Computation Technology*.

Sobral, A., & Vacavant, A. (2014). A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Computer Vision and Image Understanding*.

Stauffer, C., & Grimson, W. E. L. (1999). Adaptive background mixture models for real-time tracking. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Tang, S., Andres, B., Andriluka, M., & Schiele, B. (2015). Subgraph decomposition for multi-target tracking. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Wen, L., Du, D., Cai, Z., Lei, Z., Chang, M. C., Qi, H., Lim, J., Yang, M. H., & Lyu, S. (2020). UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. *Computer Vision and Image Understanding*, *193*.

Wen, L., Li, W., Yan, J., Lei, Z., Yi, D., & Li, S. Z. (2014). Multiple target tracking based on undirected hierarchical relation hypergraph. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Wren, C. R., Azarbayejani, A., Darrell, T., & Pentland, A. P. (1997). P finder: real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zhan, C., Duan, X., Xu, S., Song, Z., & Luo, M. (2007). An improved moving object detection algorithm based on frame difference and edge detection. *Proceedings of the 4th International Conference on Image and Graphics, ICIG 2007*.

Zhang, B., Wang, L., Wang, Z., Qiao, Y., & Wang, H. (2016). Real-Time Action Recognition with Enhanced Motion Vector CNNs. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Zivkovic, Z. (2004). Improved adaptive Gaussian mixture model for background subtraction. *Proceedings of the 17th International Conference on Pattern Recognition, 2004*, 28-31 Vol.2.