– Supplementary Material –
PathTrack: Fast Trajectory Annotation with Path Supervision

Santiago Manen[1] Michael Gygli[1] Dengxin Dai[1] Luc Van Gool[1,2]

[1]Computer Vision Laboratory
ETH Zurich

[2]ESAT - PSI / IBBT
K.U. Leuven

{smanenfr, gygli, daid, vangool}@vision.ee.ethz.ch

In this supplementary material, we present some additional information about our work. We first describe our OF-trajectory affinity measure in Sec. 1. We then present in Sec. 2 the video collection process we followed to produce the PathTrack dataset. We finalize by detailing some aspects of our experiments, including: annotation-time measurements in Sec. 3, the LP tracker that we use in our experiments in Sec. 4 and how we associate the tracklets of NOMT [b] to improve tracking performance in Sec. 5. The supplementary video shows some sequences in our dataset with their corresponding scene-label.

1. OF-trajectory affinity measure

Our affinity measure follows the intuition: detections that share many Optical Flow (OF) trajectories are likely to belong to the same object, so they should have a high affinity *c.f.* Fig. 1. To formalize this, let $\mathbf{q_i}$ be the binary descriptor of detection i. Then $\mathbf{q_{ik}}$ is the binary value that indicates whether OF trajectory k falls in detection i. And its total length corresponds to the total number of OF trajectories in the sequence. We define the affinity a_{ij} as the intersection-over-union of the binary descriptors $\mathbf{q_i}$ and $\mathbf{q_j}$:

$$a_{ij} = \frac{\left\|\mathbf{q_i} \wedge \mathbf{q_j}\right\|_1}{\left\|\mathbf{q_i} \vee \mathbf{q_j}\right\|_1} \quad (1)$$

This affinity indeed reflects whether two detections share many OF trajectories, thus being more likely of representing the same object. Note that the descriptors $\mathbf{q_i}$ only need to be computed once for each detection, making the affinity computation efficient. [c, b] also found OF trajectories useful for linking detections. In our experiments, we use the code of [c] to link the fast optical flow of [e].

2. Video collection

In this section we describe the process we followed to collect videos for the PathTrack dataset. It elaborates on Sec. 4 in the main body of the paper. Collecting a large amount of sequences for MOT is by itself challenging. The sequences should be diverse and contain scenarios interesting for tracking. Additionally, they should not contain transitions or very shaky camera movement. Taking these factors into consideration, we collected sequences for PathTrack via the following procedure:

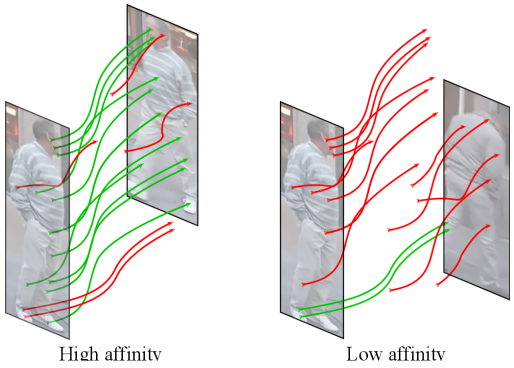

Figure 1: Detections that share many OF-trajectories are likely to belong to the same object.

1. **Video collection:** First we queried and crawled YouTube with 23 phrases corresponding to scenes with complex human behavior. These include *"pedestrian+crossing"*, *"flash+mob"* and *"basketball+game"*.
2. **Shot boundary detection:** Edited videos are usually composed of multiple shots, which is not ideal for tracking. Thus we used the sum-of-absolute-differences shot boundary detector of ffmpeg [a] to conservatively segment each video into continuous shots.
3. **Shot selection:** We manually went through the shots to select those useful as MOT sequences, i.e., those that were not too shaky and depicted complex person movement, obtaining the 720 sequences of PathTrack.

4. **Video stabilization:** Being filmed with hand-held cameras, many of the sequences were originally shaky. Shakiness complicates the annotation process and introduces an unnecessary difficulty unrelated to tracking. Therefore, we stabilized the sequences with [d].

3. Annotation time

In our experiments, we evaluate the time-efficiency of different annotation frameworks. In this section, we describe our time-measurement evaluation, which we have used to produce Fig. 6 in the main body of our work. By reviewing the working process of all the methods, we find that the time t_i to annotate a trajectory i is the summation of three components: 1) a *watching time* t_w required to follow each trajectory while annotating it, for path supervision, and for each of the N_b box annotations: 2) a *box annotation time* to annotate a bounding box t_b and 3) a *context-switching* time t_c required to understand from the *context* what correction should be made. The latter only applies to active learning methods, because they do not allow a continuous annotation flow; they jump from one frame to another abruptly and users need time to re-localize the object with the help of context information:

$$t_i = \gamma t_w + N_b(t_b + t_c) \tag{2}$$

where γ is a penalty factor that applies to our path supervision only. It accounts for the time required to watch a person in a video while following it with the mouse cursor.

We have measured these values in user study of 78 Amazon Mechanical Turk (AMT) workers and 13 experts in vision. We show in Tab. 1 our measurements. To measure the time for context-switching t_c we sequentially asked the user to annotate specific frames for specific objects according proposed by the active learning version of VATIC [i]. For each annotation, the user was allowed to browse the beginning of the trajectory and the span of time around the frame to annotate. This is necessary, since active learning usually presents challenging frames in which the object of interest is occluded or the trajectory has switched to another target. We measured how much the the user required to decide he had enough context to annotate accurately.

Table 1: Time measurements from our user study. The watching slow-down γ is only necessary for path supervision and the context-switching penalty only to the active learning version of VATIC [i].

Time component	LabelMe [k]	VATIC [j]	VATIC alearn [j]	**PathTrack (ours)**
γ	1 ×	1 ×	1 ×	**1.5** ×
t_b	5.2 s	5.2 s	5.2 s	5.2 s
t_c	0 s	0 s	**9.8 s**	0 s

Importantly, some of these factors only apply to specific methods. The video needs to be slowed down during path annotation, hence the γ_{path} of 1.5. Also, the context-switching time penalty t_c only applies to to active learning, as it is the only one that presents a frame to annotate without any other context. We summarize these differences in Tab. 1.

4. LP tracker

We detail in this section the Linear Programming (LP) tracker that we use in our experimental section, *c.f.* Sec. 5.3. We base our experiments on the standard min-cost flow formulation tracker of [l]. It formulates the Multi-Object-Tracking (MOT) problem as an integer Linear Program with 3 costs: 1) a detection-confidence cost C_i, 2) an affinity cost C_{ij} and 3) entry (C_i^{en}) and exit (C_i^{ex}) costs:

$$\begin{aligned} T = \quad & \textbf{argmin}_{\mathcal{T}} \sum_i C_i^{en} f_i^{en} + \sum_{i,j} C_{ij} f_{ij} + \sum_i C_i^{ex} f_i^{ex} + \sum_i C_i f_i \\ \text{s.t.} \quad & f_i^{en}, f_{ij}, f_i^{ex}, f_i \in \{0,1\} \\ & f_i^{en} + \sum_j f_{ji} = f_i = f_i^{ex} + \sum_j f_{ij} \end{aligned} \tag{3}$$

where the f binary variables indicate whether a cost is included in the final energy or not. The last equality is the *flow conservation constraint* that enforces the trajectory topology.

We describe now the exact form of the costs that we use for our tracker. The confidence cost C_i is $\log((1-s_i)/s_i)$, where s_i is the 0-to-1 score of detection i. Note that this cost is negative for detection scores over 0.5. This encourages using confident detections in the final tracking solution. The affinity cost C_{ij} has the form $-\log(m_{ij})$, where m_{ij} is the matching score of detections i and j. This is the score that we learn with a Convolutional Neural Network (CNN) in our experiments. Detections further than 4 seconds apart are assigned a matching score of zero, to limit big temporal jumps. The entry and exit costs have a common value of $-\log(0.1)$, as is common in tracking literature. Since we associate detections, we benefit from including the additional *context features* (e.g. relative distance, size) of [g]. These are seamlessly integrated to the matching network as input to the first fully connected layer. Similarly to the appearance-only network, training this network on our data raises the accuracy substantially, from 84% to 90%, compared to training on MOT15.

The final trajectories are the result of minimizing Eq. (3), with the aforementioned costs, using the cutting planes algorithm. For more details on this optimization, we refer the reader to [l].

5. Tracklet association

In our experiments, *c.f*. Tab. 2b, we use our matching network to improve the results of NOMT [b], the top-performing tracker in MOT15. We provide the details in this section.

Tracking methods are known to have problems tracking objects through prolonged periods of occlusion [f]. We make the observation that appearance-based matching networks can help to solve this problem.

We formulate the tracklet-association problem as a matching problem. We associate two tracklets i and j if the following conditions if: a) there matching score is at least 0.8, b) if this score is at least 5% higher than any other matching alternative, a typical *best-to-second-best* matching criteria [h] and c) if the tracklets fulfill some spatial and temporal constraints. The closest detections must be at most 4 seconds apart and at most one-fourth of the image width away. The matching score between two tracklets is computed as the median between 4 crops for each of them, uniformly sampled in a period of 2 seconds.

Our results show that we are able to reliably associate tracklets through lengthy occlusions.

References

[a] http://www.ffmpeg.org/ffmpeg-filters.html#select_002c-aselect. Accessed: November 2016. 1

[b] W. Choi. Near-Online Multi-Target Tracking With Aggregated Local Flow Descriptor. In *ICCV*, December 2015. 1, 3

[c] K. Fragkiadaki, W. Zhang, G. Zhang, and J. Shi. *Two-Granularity Tracking: Mediating Trajectory and Detection Graphs for Tracking under Occlusions*, pages 552–565. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. 1

[d] M. Grundmann, V. Kwatra, and I. Essa. Auto-directed Video Stabilization with Robust L1 Optimal Camera Paths. In *CVPR*, pages 225–232, Washington, DC, USA, 2011. IEEE Computer Society. 2

[e] T. Kroeger, R. Timofte, D. Dai, and L. V. Gool. Fast Optical Flow using Dense Inverse Search. In *ECCV*. 2016. 1

[f] C.-H. Kuo and R. Nevatia. How does person identity recognition help multi-person tracking? In *CVPR*, pages 1217–1224. IEEE Computer Society, 2011. 3

[g] L. Leal-Taixé, C. Canton-Ferrer, and K. Schindler. Learning by tracking: Siamese CNN for robust target association. *CoRR*, abs/1604.07866, 2016. 2

[h] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society. 3

[i] C. Vondrick and D. Ramanan. Video Annotation and Tracking with Active Learning. In *NIPS*, 2011. 2

[j] C. Vondrick, D. Ramanan, and D. Patterson. Efficiently Scaling Up Video Annotation with Crowdsourced Marketplaces. In *ECCV*, pages 610–623, Berlin, Heidelberg, 2010. Springer-Verlag. 2

[k] J. Yuen, B. C. Russell, C. Liu, and A. Torralba. LabelMe video: Building a video database with human annotations. In *ICCV*, pages 1451–1458. IEEE Computer Society, 2009. 2

[l] L. Zhang, Y. Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *CVPR*, pages 1–8, June 2008. 2