Multiple Object Tracking by Efficient Graph Partitioning

Ratnesh Kumar, Guillaume Charpiat, Monique Thonnat

Multiple Object Tracking by Efficient Graph Partitioning

Ratnesh Kumar, Guillaume Charpiat, Monique Thonnat

STARS Team, INRIA, Sophia Antipolis, France

Abstract. In this paper, we view multiple object tracking as a graph partitioning problem. Given any object detector, we build the graph of all detections and aim to partition it into trajectories. To quantify the similarity of any two detections, we consider local cues such as point tracks and speed, global cues such as appearance, as well as intermediate ones such as trajectory straightness. These different clues are dealt jointly to make the approach robust to detection mistakes (missing or extra detections). We thus define a Conditional Random Field and optimize it using an efficient combination of message passing and move-making algorithms. Our approach is fast on video batch sizes of hundreds of frames. Competitive and stable results on varied videos demonstrate the robustness and efficiency of our approach.

1 Introduction

The tracking of objects, persons or animals is required for high-level vision inference systems, from action recognition to animal behavior analysis. Other applications of tracking include video editing, *e.g.* inpainting, wherein a good tracking system can help in reducing the manual effort to annotate the parts of the video to keep or to remove.

In this paper, we address the problem of multiple person tracking in videos obtained from a single camera. We suppose that we are already given person detections in each frame, from any pre-trained detector. This detector is not assumed to be perfect, but may produce false negatives (persons missed) and false positives (extraneous detections). Our objective is to group detections into consistent trajectories, as well as to eliminate false negatives and false positives.

Tracking algorithms can be broadly divided into two categories: online and batch-based. Online methods infer the object state using the current frame and the previous ones; this can be achieved for instance with particle filtering [1]. On the opposite, batch-based tracking considers the entire temporal sequence at once, or at least a significant part of it (hundreds of frames). Such methods are also referred to as global tracking. Our approach belongs to the latter class, hence we focus on related work in the realm of batch-based tracking.

Global tracking should be in principle less prone to identity switches (confusion of several different persons) than online approaches, and provide better occlusion management as they have both present and future frames available

to determine the state of objects. This extra information brought by the 2D+T volume has led to the development of many different approaches utilizing all kinds of color and motion clues, as well as many different optimization methods. We discuss below some of the recent notable ones [2–11].

[4] computes cliques on a graph of detection responses to find object trajectories. False negatives and occlusions are handled using hypothetical nodes by assuming linear motion of the object. [5] presents another data association approach, based on maximum weight independent sets, to merge tracklets of length 2 to compute full trajectories. [6] uses a network flow based approach to obtain person trajectories and incorporates an explicit occlusion handling mechanism to provide information such as *object i is occluded by object j*. The work by [9] also uses a network flow based model similar to [6]. By greedily computing shortest path problems, they reach near-linear time complexity in computing the global minimum of successive sub-problems, which however doesn't guarantee a global optimum on the real global problem.

Most network flow models suffer from the disadvantage that possible occlusions or false negatives are not handled from the start, but in a post-processing step by introducing iteratively new nodes in the graph, which asks for re-iterating the optimization process until convergence. Importantly all network flow approaches and [5] do not incorporate acceleration information for computing trajectories. Acceleration knowledge is vital for ensuring smooth velocity in trajectories, and at least three nodes (*i.e.* detections) located at different time frames are required to compute acceleration. Recently a proof-of-concept work by [12] have shown that a network flow model can only deal with pairwise factors and hence cannot incorporate a third (or higher order) factor. [13] extends a typical network flow model by adding higher order connections. However in doing so they have to resort to a relaxation (discrete to continuous) based optimization scheme, as the model can no longer be minimized by efficient network flow approaches. For time critical vision applications such as tracking, a discrete to continuous relaxation approach should be avoided, as these do not provide a usable solution at all instants during their run times, and stopping the optimizer to meet time requirements might return a non usable solution.

Works by [2, 3] use a discrete-continuous optimization model to optimize detections and trajectories. They incorporate statistical data analysis to compute the parameters of the energy along with adding exclusion constraints at both trajectory and detection levels. However for this they use an already-performed tracking (from [9]) as input to their approach. Hence these works should rather be seen as trajectory-refinement process.

All of the above approaches including our proposed approach is categorized as *tracking using detections* in the literature. Instead of using detections, [8] uses *probabilistic occupancy maps* obtained from a background subtraction process. Subsequently the scene space is discretized into grid cells, and targets are forced to move along the grid. A global optimum for this model is computed using K-Shortest Paths. This scene discretization is possible only for situations wherein the camera calibration parameters are known, and hence cannot be applied to

moving camera scenarios. This approach is computationally fast but is prone to identity switches. Global appearance constraints are added by [7] to provide robustness to identity switches.

Another relevant aspect of tracking algorithms is their computational cost in practice. Most approaches performing well [2–4, 7] mention speeds of several seconds per frame. In contrast we will show an appreciable gain of speed (factor 10), with near real time performance in complicated videos. Also, [4, 7] require a lot of memory and consequently cannot handle batches of many frames (typically 50), while we do not suffer from memory consumption and will consider blocks of several hundreds of frames.

With respect to the literature, our approach has the following advantages :

- A principled way of unifying natural constraints that arise in tracking.
- Thanks to an apt combination of optimizers (section 3.1), a novel triplet search for trajectory-curvature penalization (section 2.5), and graph reduction (section 3.2), the proposed approach is computationally efficient (in terms of both memory and time), and can operate on hundreds of frames of HD videos on a simple machine.
- Last, but not least, the proposed tracker is self contained and unlike [3, 2], we do not require beforehand trajectories from an external tracker.

Following sections introduce our proposed model and the optimizers used. Subsequently section 4 elaborates on experimental results and processing times.

2 Model

A reliable tracking model should have the following criteria:

(C_1) **Uniqueness** : An object label (identity) should never appear more than once inside any frame over the whole sequence (*cf.* section 2.2).

(C_2) **Smooth Trajectories** : The trajectory of each object should be as smooth as possible in time. In particular, there should not be jumps in an object location in consecutive frames (*cf.* sections 2.2 & 2.5).

(C_3) **Robustness to Occlusions and False Negatives** : The trajectories should not be lost or confused during occlusions or when several detections are missed (*cf.* sections 2.3 & 2.4).

(C_4) **No Ghost Objects** : False Positives from the detector should be removed and not form output trajectories (*cf.* section 3.4).

The incorporation of these traits into our model will be discussed in the following sections.

2.1 Approach and Formalization

A person detector is used to obtain detection responses for all frames of a batch. We then build a graph $G = (V, E)$ whose nodes correspond to detection responses. Our aim is to find an optimal partition of G such that each part corresponds to the trajectory of one person. The notations used in the paper are presented below:

L : Number of possible labels (i.e. of graph parts).
$\mathcal{X}$: Set of all possible partitions $\subset L^{|V|}$.
i : A node of the graph, *i.e.* a person detection.
x_i : Label of the node i, to be found.
w_{ij} : Edge weight between nodes i and j (benefit for them to have same label).

For a set of L labels, each of the $|V|$ nodes can then take any of L states. A labeling $\mathbf{x} = (x_i)_{i \in V} \in \mathcal{X}$ then defines a partition of V into subsets T_l assigned to each class $l \in L$. The quality of such a labeling, to be maximized, is the sum of weighted edges connecting vertices with the same label:

$$\underset{\mathbf{x} \in \mathcal{X}}{\operatorname{argmax}} \sum_{ij} w_{ij} \ \delta_{x_i = x_j} \quad = \quad \underset{\mathbf{x} \in \mathcal{X}}{\operatorname{argmin}} \sum_{ij} w_{ij} \ \delta_{x_i \neq x_j}$$

In the following sections, we will define the quantities relevant to the problem of tracking, and show how to set the interaction terms w_{ij} to express them.

2.2 Repulsive Constraints

Adhering to the criterion C_1, it is imperative to have repulsive constraints in the model. The goal of these constraints is to prevent nodes in a same frame from being allocated the same label. To this end we set repulsive constraints by assigning huge negative weights w_{ij} between all pairs of detections (i, j) from the same frames. We refer to this as *intra-frame* repulsive constraint.
In regard to C_2, the tracker should also prevent objects from jumping from one location in a frame to a complete different location in the next frame. Otherwisely said, two nodes far apart in consecutive frames should not have the same label. To this end, we define a maximum object speed ($4m/s$) and set highly negative weights w_{ij} between detections (i, j) in consecutive frames if the speed required for such a displacement is higher. The speed between two such detections is converted from pixels/frame into m/s by multiplying them with the factor $2f/\min(H_i, H_j)$, where f is the frame rate and where H_i is the height (in pixels) of the bounding box of detection i. Indeed, $H_i/2$ is a simple yet practical and sufficient approximation of the number of pixels per meter, and it has the advantage of adapting automatically to the depth in the video. We denote these no-jump constraints as *inter-frame* constraints (*cf.* Middle Inset in Fig 1).

2.3 Temporal Neighborhoods and Point Tracks

Motion estimates for each detection provide vital information regarding its possible temporal neighbors. Without such estimates, temporal neighborhoods could become significantly big, which in turn would restrict the amount of frames processed in one batch due to a larger number of variables. Moreover, detections may be infrequent (false negatives) and may contain false positives, which prevents obtaining good motion estimates.

Owing to these reasons we do not rely only on the location of the detections, but instead we use point tracks, which provide motion estimates on a pixel

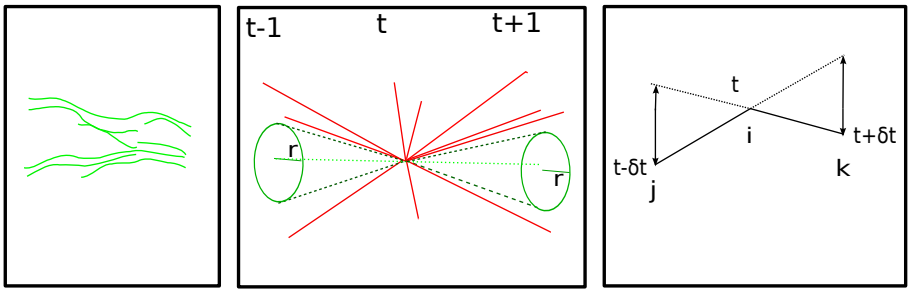

Fig. 1. Left: Point Tracks and detections. **Middle**: High repulsive exclusion constraints, shown for the central green bounding box in frame t, with **red** edges. Intra-frame repulsive constraints prevent this detection from having the same label as any other detection in the same frame t, while inter-frame repulsive constraints prevent it from having the same label as detections in previous and next frames $t-1$, $t+1$ that are too far. The maximum radius **r** is detection-dependent, in that it corresponds to the maximum physically-plausible displacement within a duration of one frame, and that speed estimation depends on object depth. **Right**: Computing the trajectory straightness for a triplet of detections j, i, k at the temporal scale δt (*cf.* sec 2.5).

(or sub-pixellic) level. The proportion $F(i,j)$ of common point tracks between two nodes i and j, i.e. of points tracks which go through two given bounding boxes B_i and B_j from different times, defines a first similarity measure between these nodes. This similarity measure has the advantage of being robust to false detections (positive as negative) that occur between the two nodes. Also, in the case of partial occlusions, point tracks on the non-occluded part will still manage to follow the object and define a meaningful similarity, while overlapping detections make appearance-based affinities noisy in such partial occlusion cases.

2.4 Appearance Connections

In the previous sections, apart from defining repulsive constraints, we have added connections between the nodes which share common point tracks, as an incentive for them to choose the same label and consequently be part of the trajectory of the same object. But point tracks most often do not cover the full temporal span of the video and tracks may get lost or confused (aperture problem) after some time due to occlusion or pose changes. This calls for the need of *long temporal range connections* in the graph. To define such a similarity between detections from any frames, possibly far apart in time, we consider the appearance of objects. To cater this, we compute an appearance feature for each bounding box, and cluster these appearances into L groups using K-means. The appearance cluster of node i is denoted by A_i. The appearance similarity between any nodes belonging to a same cluster is then defined as a constant weight $\beta > 0$, while the one between nodes from different cluster is 0, which will lead to the quantity $\beta\ \delta_{A_i=A_j}$ in Equation (3). For appearance clustering we do not need to know the exact number of objects in the scene, as over-clustering is not an issue with

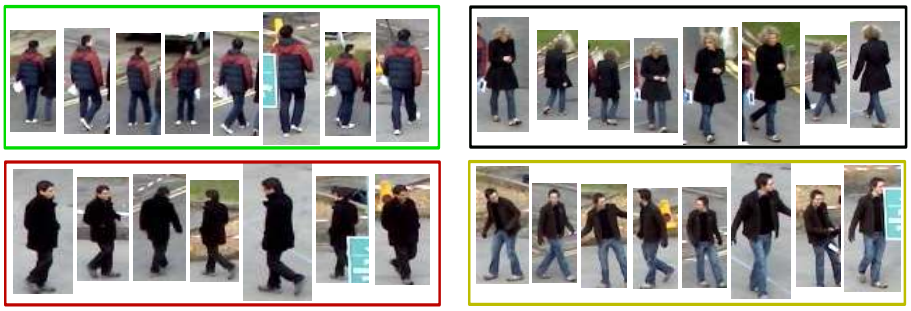

Fig. 2. Sample appearance clusters for PETS09 S2L1: Each box encloses a few samples belonging to the same cluster. Notice that the clustering is robust to scale changes.

our approach. In our experiments we set L to twice the maximum number of detections observed in a single frame of the current video batch.

The appearance feature for a detection is based on the histogram of colors inside the detection The distance between two appearances is computed by the L_2 norm between features. Fig 2 shows a few clusters from a PETS09 [14] video.

2.5 Favoring Smooth Trajectories

When the trajectories of people with a similar appearance cross each other, and that, in plus, full occlusion or detection mistakes (false positives or negatives) occur at this crossing, then the tracking based on the previous quantities may fail and induce small jumps in location or identity switches. This can be mitigated by *penalizing high curvature* of the trajectories. To this end we define an energy term which favors smooth trajectories, involving three detections j, i, k from different frames regularly spaced in time $(t - \delta t, t, t + \delta t)$, as shown in Right-most inset of Fig 1. Given such detections, we compute a triplet factor $R(i, j, k)$ expressing the regularity of the associated trajectory. It is based on the Laplacian of the centroids p_i of the three bounding boxes:

$$\Delta_{ijk} \;=\; \frac{f}{H_i\ \delta t}\ \ \|p_j + p_k - 2p_i\| \tag{1}$$

and is expressed as:

$$R(i,j,k) \;=\; \frac{1}{\sqrt{\delta t}}\ \ \max(\tau - \Delta_{ijk},\ \ 0) \tag{2}$$

Note that the Laplacian was normalized in order to be invariant to the video resolution and to the time interval, using the frame rate f, the frame interval δt and the bounding box height H_i used as previously as a depth/resolution indicator. R denotes the benefit for the triplet j, i, k to belong to a same trajectory. The threshold τ indicates the maximum speed difference, or curvature, above which there is no gain. Note that $\tau = 1m/s^2$ is video-invariant.

Considering all triplets in the graph is computationally prohibitive as there are $|V|^3$ possible triplets ($|V|^2$ symmetrical triplets). Furthermore searching only temporally successive triplets ($\delta t = 1$) would be sensitive to missed detections. Thus we compute $R(i,j,k)$ on various temporal scales, with $\delta t \in \{1,2,3,4,5,10\}$. As object trajectory is more predictable at shorter temporal ranges, we decrease the importance of the smoothness assumption for longer time spans with the factor $1/\sqrt{\delta t}$.

Unlike [3], which requires the knowledge of trajectory proposals for curvature penalization, we model curvature penalization using an apt triplet search, thus making our model as unified and self contained as possible.

3 Optimization

Summing up all quantities defined in the previous section, we obtain the following similarity criterion to maximize over possible labellings $\mathbf{x}$:

$$\begin{aligned} S(\mathbf{x}) = \alpha \sum_{ij} F(i,j)\ \delta_{x_i=x_j} \quad + \quad \beta \sum_{ij} \delta_{A_i=A_j}\ \delta_{x_i=x_j} \\ +\gamma \sum_{ijk} R(i,j,k)\ \delta^T_{x_i,x_j,x_k} \ - \ \Omega \sum_{(i,j)\in\mathcal{C}} \delta_{x_i=x_j} \end{aligned} \tag{3}$$

where $\delta_{\text{true}} = 1$ and $\delta_{\text{false}} = 0$, where $\delta^T_{x_i,x_j,x_k} = \frac{1}{3}\left(\delta_{x_i=x_j} + \delta_{x_i=x_k} + \delta_{x_j=x_k}\right)$ is a good 2nd-order approximation of $\delta_{x_i=x_j=x_k}$. $\mathcal{C}$ denotes the set of all pairwise inter-frame or intra-frame constraints, and $\Omega > 0$ is sufficiently high to be dissuasive. Maximizing $S(\mathbf{x})$ in (3) is equivalent to minimizing $E(\mathbf{x}) = -S(\mathbf{x})$. Note that if one denotes by $\mathcal{X}$ the set of feasible labellings, i.e. satisfying all constraints, then solving $\operatorname{argmin}_{\mathbf{x}\in L^V} E(\mathbf{x})$ with high Ω is equivalent to solving $\operatorname{argmin}_{\mathbf{x}\in\mathcal{X}} E(\mathbf{x})$ with $\Omega = 0$, and that $E(\mathbf{x})$ in this latter formulation is submodular on $\mathcal{X}$, but to our knowledge there is no optimizer dedicated to the minimization of submodular functions over arbitrary sets.

3.1 Optimizers

The criterion (3) is a non-submodular Conditional Random Field, and a host of available optimization techniques can be employed. This section aims at providing a concise overview on selecting optimizers for $E(\mathbf{x})$.

Heuristic approaches such as Tabu Search (used by [4]) or Simulated Annealing can be used to optimize (3). However these suffer from at least two notable problems to applied to a tracking model. Firstly, they do not provide any indication certifying the vicinity of a solution from optimal. Secondly, in time critical applications it is often desirable to use techniques which can provide a usable solution (*i.e.* a solution satisfying all constraints) whenever required. These heuristic approaches along with relaxation techniques (*e.g.* Lagrangian relaxation), Polyhedral methods, do not provide a usable solution at all time instants of their operation. In a recent work by [15], the authors show that

combinatorial methods (such as Integer Linear Programming, Max Cut using Reweighted Perfect Matching) based on branch-and-bound and cutting plane techniques do not scale well with the problem size.

Therefore we employ a fast message passing variant: Tree Re-Weighted Message Passing (**TRW-S**) [16]. TRW-S considers the full graph for labeling and provides good optima (along with the optimality bound). To make further improvements, we design local moves based on an Iterated Conditional Modes (ICM) that will satisfy the constraints and are likely to be useful.

ICM [17] is an iterative procedure, wherein an optimum labeling for a variable is chosen conditioned on all other variables. The process is repeated for all variables until convergence. Owing to this local nature of moves it is practically difficult for ICM to find a usable solution from scratch. However with a good initialization, ICM can find more meaningful solutions quickly. [18] proposed a variant of ICM: Lazy Flipper (**LF**). As opposed to ICM, a LF takes into account a large connected subset of variables up to a prescribed size. Upon convergence, the LF's labeling is guaranteed to be optimal within a Hamming distance equal to the subgraph size.

Inspired from LF, we search for flips of variables which reduce the energy, by considering a single variable or a subset of variables at a time. For a given subset size we repeat the above process until the energy cannot be reduced further. We also limit our search to only those subsets of variables which have a joint temporal distance of less than 20 frames, i.e. the maximum time scale in triplets. This structural information incorporation brings significant computational speed up on this part of the optimization process.

Notice that this flipper is different from the Block-ICM proposed by [12]. The authors in [12] sweep in a temporally forward manner, and check for flips *locally* in two frames. The flipper in our approach is more general as the nodes considered for flips have no time ordering restrictions. Furthermore as an ICM is susceptible to initialization ([12]), we provide a good usable solution satisfying all constraints to our flipper, obtained from TRW-S.

We experimented with recently proposed optimizers based on dual decomposition [19] and linear programming relaxations [20]. Both these techniques were excruciatingly slow on finding one usable solution.

3.2 Graph Reduction

Point tracks can be used to reduce the initial graph, and consequently speed up the optimization, by fusing nodes for which there is no tracking ambiguity. More

Fig. 3. Graph reduction by fusing nodes u,v and w,x.

precisely, two nodes are fused if *all* point tracks inside both bounding boxes

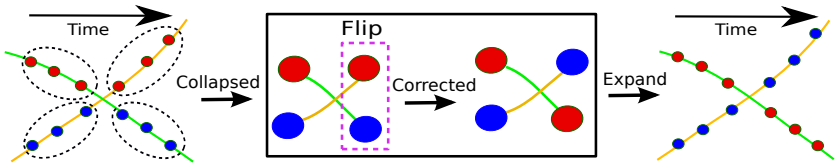

Fig. 4. Graph reduction and flipper speed gain. Leftmost drawing shows the labeling of 12 nodes. Node colors indicate a labeling (stuck at a local minima) and the colored lines (*green* and *yellow*) correspond to the groundtruth trajectories. Middle inset drawing shows the reduced graph by fusing nodes inside the *dotted* ellipse in the Leftmost drawing (*cf.* sec. 3.2). Since the graph is reduced, we only need to search for flips of subgraphs of size = 2 (*i.e.* $O(|V|^2)$), instead of enumerating over all possible subgraphs of size = 6 (*i.e.* $O(|V|^6)$). The subgraph-flip required to correct the initial labeling is shown by *magenta* colored rectangular box in the middle inset. The Rightmost drawing correspond to the final correct output in the original graph.

move into one another. When fusing nodes i and j, their edge weights (towards any other node k) are accumulated, as in Fig 3 : $w_{\{ij\},k} = w_{ik} + w_{jk}$. Thus the criterion optimized does not change.

The ratio of the fused graph size to the original size translates directly to the optimization speed, as we observe a reduction in the computational cost by the same ratio. Fig 4 shows a simple illustration for the speed gain due to this graph reduction step (*cf.* Fig 4 caption for details).

3.3 Parameter Setting

We have three constant factors in CRF, namely α, β and γ (*cf.* equation (3)). These have to be set considering the relative importance of each term, and this setting should be as invariant to videos as possible. We set $\Omega = 1000000$, sufficiently high to enforce repulsive constraints. The rationale in setting these parameters is outlined below.

- The main contribution of point tracks is to obtain motion estimates which help in fusing the graph. On the fused graph, the nodes are either close to occlusion vicinity or false negatives. In both these situations, we need to look at appearance and trajectory curvature in time to decide about the labeling. Hence we set more weight on triplets and appearance clusters.
- Near occlusion vicinity, it is important to consider both motion difference and appearance. However trajectory curvature in short temporal scale should have higher weight than appearance due to following reasons: Firstly, trajectories are not supposed to change their path substantially in course of a few frames, and secondly, the appearance of persons can be similar either due to the presence of noise or persons having the same appearance. Hence $\beta = 0.5$ is set lower than $\gamma = 50$, with $\alpha = 0.01$. A small perturbation of these values while respecting the relative order does not change the results.

3.4 Graph Cleaning

After solving the optimization on a batch, any temporal gap along a trajectory due to false negatives is filled by assuming linear motion between the closest detected bounding boxes on either side of the temporal axis. False positives from detections usually form short trajectories and with long temporal gaps, and these tracks are removed.

Streaming Trajectory Computation is performed using a sliding window approach, with an overlap of 15 frames from the previous batch.

4 Experimental Results and Evaluation

Comparative quantitative evaluation is challenging due to the lack of benchmark datasets, such as [21] for optical flow evaluation purpose. For our evaluation purposes, we use the metrics proposed by [6][1]. Recently, [22] has thrown light on the ambiguities in evaluation due to dissimilarities in groundtruth annotations and evaluation codes used by different tracking approaches. Taking these ambiguities into account we provide other relevant information, *e.g.* type of detector, groundtruth and the code used for computing the metrics.

4.1 Evaluation Metrics

- **Recall**: % of correct detections w.r.t. total detections in groundtruth.
- **Precision**: % of correct detections w.r.t. total detections in tracking.
- **FAF**: False alarms per frame. **GT** : Number of trajectories in groundtruth.
- **MT**: Number of trajectories successfully tracked for $> 80\%$ of their groundtruth time span. **ML** : Number of trajectories successfully tracked for $< 20\%$ of their groundtruth time span.
- **IDS**: Number of times a tracked trajectory changes its matched id.

Other widely adopted metrics are Multiple Object Tracking Accuracy/Precision (MOTA and MOTP) [23]. MOTA takes into account the number of false positives, false negatives and IDS. MOTP quantifies the average distance between the groundtruth and the estimated target locations. MOTA is the preferred metric for us as we do not change the locations of input detections.

4.2 Results and Discussion

We consider videos from public datasets for evaluation. TRW-S is run for a fixed number of iterations (50), and the output is then optimized by the flipper. **All our results are obtained with the same unique set of parameters.**

PETS dataset: We use the S2L1 view. This video is challenging due to non linear motion, and persons of similar appearances occluding each other [2]. With global appearance features, we are able to keep the same ID for persons exiting and re-entering the scene (*cf.* Fig 5).

[1] We use the code provided by [2] to compute these metrics.
[2] Detections from http://iris.usc.edu/people/yangbo/downloads.html

Method	Rec	Prec	FAF	GT	MT	ML	IDS	MOTA(3D)	MOTA(2D)
Milan *et al.* [3]	96.9	94.1	0.36	19	18	0	22	90	83.6
Shitrit *et al.* [7]	-	-	-	-	-	-	19	-	82
Berclaz *et al.* [8]	-	-	-	-	-	-	28	-	80
Zamir *et al.* [4]	-	69.4	-	-	-	-	8	-	90.3
Ours	95.8	90.8	**0.28**	19	18	0	13	**90**	81.8

Table 1. Comparison with the recent proposed approaches on PETS S2L1 Video. The metrics on the first and last row are obtained using the same code, while middle ones are taken from the respective papers as their result files are unavailable.

We observe (Table 1) that our approach performs well and is on par or better with the best in the metrics. At the same time our approach is much faster (*cf.* section 4.3) and **do not rely** on external tracker as [3] or camera calibration as [3, 7, 8]. Also the model of [3] has 11 parameters and requires an explicit parameter estimation step for a video. [4] uses Tabu Search on their model and hence guarantees of a stable output on multiple runs are minimal. Moreover [4] do not cater to a proper video streaming, as merging trajectories from batches is based on solving a clique problem by considering multiple batches. Typically better cliques can be found if more batches are considered for merging, which in turn adds to the latency time.

Usefulness of Triplets: Triplets are necessary to maintain consistent identities in time. Indeed without using *triplets*, our results for MOTA (3D) would decrease to 84.4 and the *identity switches* would increase to 35.

Towncenter dataset: For the evaluation on HD videos with higher person density, we consider Towncenter video [11]. The groundtruth, result and detection files can be obtained from the webpage of [11]. We achieve competitive MOTA[3] (*cf.* Table 3, Fig 7). [11] uses a head detector along with motion estimates to make precise the location of body and head bounding boxes, hence achieves higher MOTP. Also [11] requires camera calibration and a parameter estimation step which takes several hours.

Although [24] is a frame-by-frame method, we compare to this approach as the output files are available, and as it outperforms other previous approaches. While being competitive, we are 6 times faster on a single core implementation than the dual core implementation of [24].

Optimizer Scaling w.r.t. problem size: With this proposed combination of optimizers on a batch of 400 frames (with a high density of 15 persons/frame), we achieve competitive MOTA as shown in Table 2. The maximum memory consumed by our algorithm is under 6 GB, demonstrating the memory efficiency.

Stable output on multiple runs: We performed 5 trials on 50 frame batches for the first 1000 frames, and obtain a stable MOTA with a standard deviation of 0.7. In these trials we stop the optimizer to meet a requirement of **5-10 fps** (by stopping the flipper). Such stability guarantees can not be provided by heuristic (or stochastic) optimization approaches *e.g.* Tabu Search.

[3] MOTA code from https://github.com/glisanti/CLEAR-MOT

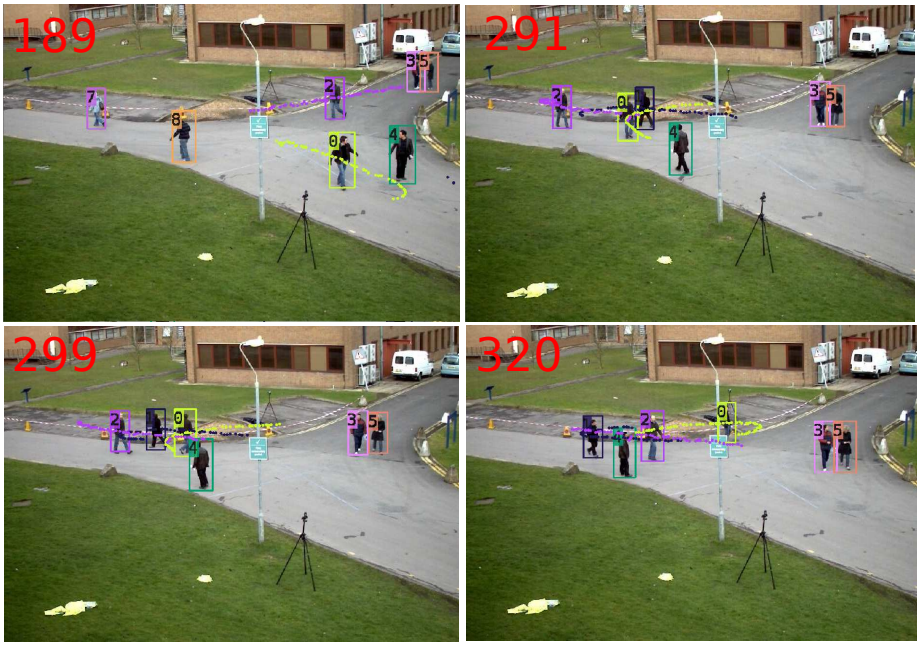

Fig. 5. Consistent tracks before and after occlusion: Frame numbers are shown on Top-Left of each image. The dotted points show the trajectories of bounding boxes in previous and successive frames. To reduce clutter in display, we show 3 trajectories and their *few* previous and future locations. Owing to the global appearance incorporation, **ID 4 is kept intact** for the person as he leaves and comes back in the view. This immaculate consistency in identities during crossing is obtained due to *triplet factors*.

Towncenter			
Method	MOTA	MOTP	Detector
Benfold *et al.* [11]	55.8	84.4	[11]
Zhang *et al.* [24]	66.4	75.2	[11]
Ours	66.5	75.8	[11]

Table 2. Results on a crowded batch of 400 frames. There are 6000 detections in this batch starting at the 450th frame.

Towncenter			
Method	MOTA	MOTP	Detector
[11]	55.9	84.7	[11]
[24]	64.9	76.4	[11]
Ours	64.6	75.0	[11]

Table 3. Results for first 1000 frames (50 frame batches). Competitive accuracy is obtained at **6x** faster speed.

.

Usefulness of Point Tracks: Point tracks are an integral part of many vision systems. For our tracking system, point tracks help in reducing the graph and in-turn aid in swift convergence of our optimizer. On standard GPUs, optical flow based point-tracker such as [25] and the KLT based point-tracker from [26] provide dense point tracks at 25 fps and 200 fps, respectively for high resolution videos. Moreover as ours is a batch based tracker, a simple dual-threaded implementation with a dedicated thread for computation of point tracks for the successive batch can remove any probable latency. In order to demonstrate the advantage of graph reduction, we consider the complete batch of 794 frames of

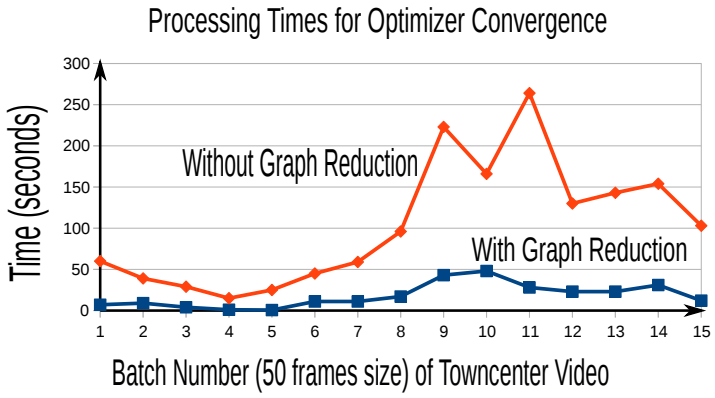

Fig. 6. Processing time improvement using graph reduction (similar MOTA in both cases), for the first 15 batches. High processing times from batch 8 to 12 are due to high detection rate/frame. In the above cases we let the optimizer run until convergence.

the PETS S2L1 video, with or without the graph reduction step (*cf.* Table 4). We observe a **4x** increase in efficiency with the graph reduction step. Similar comparative details on Towncenter video can be seen in Fig 6.

Batch Size (Number of Frames)	*App*+*k*-means	TRW-S	Flipper	Total	Memory
794 (With Graph Reduction)	35	95	370	**500**	2.5 GB
794 (Without Graph Reduction)	35	350	1500	**1885**	5 GB

Table 4. Usefulness of graph reduction: PETS S2L1 computation times (in **seconds**) for the cases when the graph reduction step is either chosen (*Row 1*) or discarded (*Row 2*). The column ***App*+*k*-means**, shows cumulative times for both appearance feature computation and clustering. The quantitative results for both rows are similar.

4.3 Processing Time

Computational times are one of the major factors in choosing a tracking algorithm. Comparing exact running times is challenging, as peers often do not provide them (nor the code) or quote only average times over a set of varied videos. Our optimizer takes 0.01 s/frame on average for 50 frames batches (8 persons/frame, PETS S2L1) on a **single 2.4 GHz core**, which is **100x** faster than the best case of [3] in 2013. The run time of optimizer from [3] is 1-2 s/frame. The average run time for [4] optimizer is 4.4 s/frame on **four** 2.4 GHz cores. On videos similar to PETS S2L1, [7] takes 4s/frame on 3 GHz CPU. Table 5 shows computational efficiency of our approach w.r.t. recent approaches.

We perform optimization at a speed of 5-11 fps (on 50 frames batches the Towncenter with 10-17 persons/frame), which is **20x-40x** faster than the 4-core implementation of [4], and **6x-10x** faster than the 2-core implementation of [24]. [11] achieves a real time performance on Towncenter using parallel CPU implementation, however the tracking accuracy is significantly lower than us.

Fig. 7. Consistent people crossing over a span of **134** frames. The above images are cropped to show the crossing.

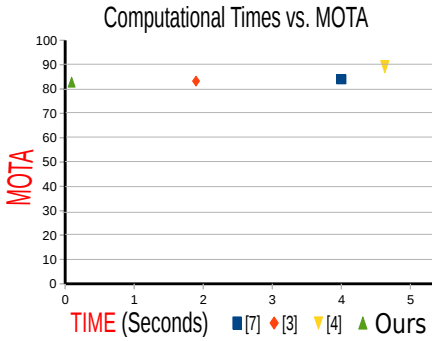

Table 5. Approximate comparison of computation times for PETS S2L1, showing a significant positive gap between ours and state-of-the-art approaches in terms of speed *vs.* accuracy.

Method	Optimization	TOF	GAF
[9]	K-Shortest Paths	×	×
[4]	Tabu Search	×	✓
[3]	α-Expansion, TRW-S	✓	×
[11]	Markov Monte-Carlo	×	×
[7]	Integer Programming	×	✓
[13]	Lagrange Relaxation	✓	×
Ours	TRW-S, Flipper	✓	✓

Table 6. State-of-the-art approaches and their incorporation of local and global clues. TOF stands for 3rd order factor (triplet). GAF denotes global appearance factor.

5 Conclusion

We proposed a novel tracking algorithm which exploits local and global cues in the most unified manner as compared to the state-of-the-art (*cf.* Table 6). Furthermore an apt combination of optimizers and suitable graph reduction lends robustness, stability and efficiency (5-10 fps on HD video of busy town street, on a single core CPU) to the model. The results obtained are competitive or better than the state-of-the-art at this speed. The proposed model has two degrees of freedom, and hence would be highly amenable to parameter learning using scene context and other relevant inputs. Also the algorithm is easy to implement and reproduce. Future prospects include tracking object parts and providing feedback to the point-trackers about incorrect tracks.

Supplementary materials can be found at the project webpage.[4]

Acknowledgements: This work has received funding from the European Community's FP7/2007-2013 - under grant agreement n° 248907-VANAHEIM.

[4] http://www-sop.inria.fr/stars/Documents/tracking/

References

1. Breitenstein, M., Reichlin, F.: Robust tracking-by-detection using a detector confidence particle filter. In: ICCV. (2009)
2. Andriyenko, A., Schindler, K., Roth, S.: Discrete-continuous optimization for multi-target tracking. In: CVPR. (2012)
3. Milan, A., Schindler, K., Roth, S.: Detection-and Trajectory-Level Exclusion in Multiple Object Tracking. In: CVPR. Number June (2013)
4. Zamir, A., Dehghan, A., Shah, M.: GMCP-Tracker: Global Multi-Object Tracking Using Generalized Minimum Clique Graphs. In: ECCV. (2012)
5. Brendel, W., Amer, M., Todorovic, S.: Multiobject tracking as maximum weight independent set. CVPR (2011)
6. Zhang, L., Li, Y., Nevatia, R.: Global data association for multi-object tracking using network flows. In: CVPR. (2008)
7. Shitrit, H.B., et al. Berclaz, J.: Tracking multiple people under global appearance constraints. In: ICCV. (2011)
8. J. Berclaz, F. Fleuret, E.T., Fua, P.: Multiple Object Tracking using K-Shortest Paths Optimization. TPAMI (2011)
9. Pirsiavash, H., Ramanan, D., Fowlkes, C.C.: Globally-optimal greedy algorithms for tracking a variable number of objects. CVPR (2011)
10. Russell, C., Agapito, L., Setti, F.: Efficient Second Order Multi-Target Tracking with Exclusion Constraints. BMVC (2011)
11. Benfold, B., Reid, I.: Stable multi-target tracking in real-time surveillance video. In: CVPR. (2011)
12. Collins, R.T.: Multitarget data association with higher-order motion models. In: CVPR. (2012)
13. Butt, A., Collins, R.: Multi-target tracking by Lagrangian relaxation to min-cost network flow. In: CVPR. (2013)
14. Ellis, A., Shahrokni, A., Ferryman, J.: PETS2009 and Winter-PETS 2009 results : A combined evaluation. In: PETS Workshop, Ieee (2009)
15. Kappes, J.H., Speth, M., Reinelt, G., Schnorr, C.: Towards Efficient and Exact MAP-Inference for Large Scale Discrete Computer Vision Problems via Combinatorial Optimization. In: CVPR. (2013)
16. Kolmogorov, V.: Convergent tree-reweighted message passing for energy minimization. TPAMI (2006)
17. Besag, J.: On the Statistical Analysis of Dirty Pictures Julian Besag. Statistical mehodological Society (1986)
18. Andres, B., Kappes, J.H., Kothe, U., Hamprecht, F.A.: The Lazy Flipper : MAP Inference in. In: ECCV. (2012)
19. Martins, A.F.T., Figueiredo, M.A.T., Aguiar, P.M.Q., Smith, N.A., Xing, E.P.: An augmented Lagrangian approach to constrained MAP inference. In: ICML. (2011)
20. Sontag, D., Choe, D., Li, Y.: Efficiently searching for frustrated cycles in MAP inference. In: Uncertainty in Artificial Intelligence. (2012)
21. Butler, D.J., Wulff, J., Stanley, G.B., Black, M.J.: A naturalistic open source movie for optical flow evaluation. In: ECCV. (2012)
22. Milan, A., Schindler, K., Roth, S.: Challenges of Ground Truth Evaluation of Multi-Target Tracking. In: CVPR Workshops. (2013)
23. Bernardin, K., Stiefelhagen, R.: Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. EURASIP JIVP (2008)

24. Zhang, J., Presti, L., Sclaroff, S.: Online multi-person tracking by tracker hierarchy. In: AVSS. (2012)
25. Senst, T., Eiselein, V., Sikora, T.: Robust local optical flow for feature tracking. IEEE Transactions on Circuits and Systems for Video Technology (2012)
26. Zach, C., Gallup, D., Frahm, J.: Fast gain-adaptive KLT tracking on the GPU. In: CVPR Workshops. (2008)