Improving tracking with a tracklet associator

Rémi Nahon, Guillaume-Alexandre Bilodeau and Gilles Pesant
Department of Computer Engineering and Software Engineering
Polytechnique Montréal, Montréal, Canada
nahon.remi@polymtl.ca, gabilodeau@polymtl.ca, gilles.pesant@polymtl.ca

Abstract—**Multiple object tracking (MOT) is a task in computer vision that aims to detect the position of various objects in videos and to associate them to a unique identity. We propose an approach based on Constraint Programming (CP) whose goal is to be grafted to any existing tracker in order to improve its object association results. We developed a modular algorithm divided into three independent phases. The first phase consists in recovering the tracklets provided by a base tracker and to cut them at the places where uncertain associations are spotted, for example, when tracklets overlap, which may cause identity switches. In the second phase, we associate the previously constructed tracklets using a Belief Propagation Constraint Programming algorithm, where we propose various constraints that assign scores to each of the tracklets based on multiple characteristics, such as their dynamics or the distance between them in time and space. Finally, the third phase is a rudimentary interpolation model to fill in the remaining holes in the trajectories we built. Experiments show that our model leads to improvements in the results for all three of the state-of-the-art trackers on which we tested it (3 to 4 points gained on $HOTA$ and $IDF1$).**

Keywords-**Multiple Object Tracking; Constraint Programming; Belief Propagation; Tracklets**

I. Introduction

Multiple object tracking (MOT) aims to detect and affect a unique identity to various objects in video sequences. It is in many cases solved by the *tracking-by-detection* paradigm, which consists in separating the problem into two distinct tasks, detection and association (which we do in this work), but can also be solved by *tracking-by-regression*, a method that performs these two actions in parallel for short-term associations. While recent advances in machine learning (ML) have led to a huge performance gain for the detection phase in MOT, the association phase remains a challenge, especially because of its combinatorial complexity.

In this paper, to better deal with the combinatorial complexity in the association phase, we propose a module based on Constraint Programming (CP) whose goal is to be grafted to any existing tracker in order to improve its object association results. It can be applied to methods that are from either the *tracking-by-detection* or *tracking-by-regression* framework since our method addresses long-term data association at the tracklet level.

Our proposed model is divided into three independent phases. The first phase, called *TrackletCutter* consists in recovering the tracklets provided by a base tracker and to cut them in places where uncertain associations are spotted. In the second phase, called *CP Associator*, we associate the previously constructed tracklets using a Belief Propagation Constraint Programming model, where we propose various novel constraints that assign scores to each of the tracklets based on multiple characteristics, such as their dynamics or the distance between them in time and space. Finally, the last phase is a rudimentary interpolation model to fill in the remaining holes in the trajectories we built.

In the experiments, we show the benefit of our method with improvements in the results for all three of the state-of-the-art trackers on which we have tested it (3 to 4 points gained on the *HOTA* and *IDF1* metrics).

II. Background and related work

MOT is a fast-growing field with many existing approaches whose performance has been greatly improved by the recent advances in machine learning. It can mostly be divided into two aspects (that are most often phases of resolution) : the object detection and the object association.

A. Object Detection

ML detection techniques have recently allowed great advances in the field of object detection. This is the case, for example, of the detectors of the *R-CNN* family [1] whose principle consists in extracting regions of interest (ROI) and then, via convolutional neural networks ($CNNs$), in inferring the main features of these ROI in order to find the class of an object and its exact position in the image. Improvements have been made to this method, in particular with the Faster *R-CNN* detector ($FRCNN$) [2] for which the region proposal method is itself a neural network called *Region Proposal Network*. The typical output of a detector is usually provided in a (x, y, w, h) format (where (x, y) represent the spatial coordinates of the upper-left corner of the bounding box and (w, h) its width and height). Such detections are typically the inputs of trackers. Since detection is not

Figure 1. Typical tracking result for a frame in the MO17-09 sequence : each of the pedestrians is enclosed by a bounding box whose color corresponds to an identifier that should be kept by the same pedestrian during the entire sequence.

the subject of this article, we can just remember here that better detections lead to better tracking.

B. Object Association

Given a set of detections at each frame, a *MOT* method typically builds tracks by associating detections across frames. The tracker aims at affecting a single identifier to each object of interest. To efficiently perform the association phase, two questions arise:

1) How to represent the detections in such a way as to be able to recognize and differentiate the different objects of interest?
2) How to efficiently explore the set of solutions and reach a satisfactory solution in a reasonable time?

1) Models to describe the detections: Regarding the representation of detections, for an efficient association, we generally seek to maximize one (or more) similarity metric between associated detections, and these metrics are computed from descriptors that are mainly divided into motion and appearance models.

Motion models are generally based on the fact that the objects of interest being tracked meet a certain number of physical constraints concerning their speed of movement and deformation in the image. Because videos are often captured at more than 30 frames per second, the displacement in pixels of the classes of objects of interest is normally small between frames. We can consider what we will call *positional models* which make the hypothesis that at the scale of successive frames, the object of interest followed is immobile, and of fixed size and shape. This allows the design of very simple and therefore very fast models such as the one on which much of our study is based, *IOU-Tracker* [3], which consists in looking for the best associations of detections frame by frame by maximizing their *IOU* (quotient of their intersection and union surfaces). This metric measures the superposition of two detections, and penalizes the differences in size and position. To increase the accuracy of this kind of *positional model*, one can consider that the derivatives of these position and size values are fixed and use these to

predict the position of the object in subsequent frames. This is the principle of the *Kalman filter* [4] used for example in *SORT* [5], one of the trackers on which we test our model. The tracking task can be simplified by keeping a model of the camera movement in parallel with the movement models of the objects of interest [6].

Appearance models consist in the description of some visual characteristics of the objects of interest (or rather of their bounding boxes) and are based on the fact that an object of interest generally keeps a similar appearance through time (or changes only slightly and progressively anyway), which will allow to associate the detections with similar descriptors. These descriptors are generally based on the distribution of colors [7], gradients of intensities [8] or more advanced methods such as covariance matrices and multiple kind of filters [9]. *CNNs* can also be used to model appearance [10], as well as transformers [11] and Mixture Density Neural Networks [12]. The method proposed by *Tracktor* [13], on which we test our model, consists precisely in a regression of bounding boxes from one frame to another to extend the trajectories. *CenterTrack* [14] works in a similar way but by working on the center of the detections, which allows it to be more robust to occlusions. For all these methods the regression is used for short-term association.

2) Resolving the association problem: Once the representation models have been chosen, we try to associate detections with one another to build our trajectories. The goal is then to assign a trajectory identifier to each detection to ensure that we can track every object of interest from its entry into the camera field of view (*FOV*) to its exit of it. Trackers can be divided into two categories regarding the way they process data.

Online trackers [3], [5] aim to build their trajectories in real-time and therefore work frame by frame in chronological order. The method they use must thus be incremental and build up tracks by adding detections to existing tracks at each new frame. The tracker must have a criterion to open a track (i.e. state that a new object entered the FOV), to close one (i.e. state that it has left the FOV) and to add a detection to a track. These kinds of trackers can only be based on information from past trajectories and current detections (that provide only static information) and cannot return to modify previously constructed trajectories.

Offline trackers remove the real-time constraint and can thus be applied to the whole set of detections at once, and our work fits into this paradigm as it allows to consider the association problem as a global optimization problem. Even if it allows access to many new characteristics of the objects of interest and modes of resolution, this shift to *offline* greatly increases the complexity of the problem. Different avenues are taken to resolve that difficulty. Some methods resolve it by

still working frame by frame, whether it is with dynamic programming [15] or by going through time in both directions [16] to make their results more robust than they would be online. Some choose to go from a local to a global scale, as in the *H2T* tracker [17], which divides frames into small subsets, minimizes a sum of affinity functions to associate detections within them, and then resolves recursively the same problem on bigger and bigger associated sets until whole tracks are finalized.

III. APPROACH AND METHODOLOGY

A. Motivation

We build on the work of [18] who was the first to apply *CP* to multi-object tracking. The benefit of choosing *CP* is that it is a very efficient method of formalizing and solving optimization problems [19]. This is the reason why we have decided to pursue in that direction. The choice of working on the association of pairs of detections was the main limitation of this previous work. Indeed this made the search space extremely vast and therefore the computation time quite high. This forced the decision to work in batches of frames and to strongly restrict the spatial distances between bounding boxes considered, which increases the sensitivity of the model to occlusions.

To remedy this difficulty, our main idea is to work at the level of tracklets (sequences of detections) instead of individual detections. This greatly reduces the size of the problem. Indeed there are simple trackers that are extremely efficient in terms of execution time (notably *IOU-Tracker* [3] that manages to process up to 100,000 frames per second, making it hundreds of times faster than most of the other state-of-the-art trackers) and they are good at making associations in simple cases. We decided to start from them to capitalize on their speed and try to correct their errors a posteriori, especially those which intervene in the case of occlusions. This choice also allows us to increase the refinement of the association model, which can now be based on the characteristics of the tracklets, which are dynamic, and not only on those of the detections, which are static.

Our model is also applicable to all kinds of trackers since we work from their outputs. Here we will speak of *tracklet* for any association of detections provided by an initial tracker, whether we have cut it or not, and of *trajectory* to speak either of the associations of tracklets that we constructed, or of the associations of detections provided by the ground truth representing a single object.

B. A three-phase model

Our model is divided into three modules :

- the **TrackletCutter** cuts the tracklets provided by the initial tracker where they intersect each other with a high degree of overlap;
- the **CP Associator** is the original association model we propose here, based on a Belief Propagation Constraint Programming algorithm [20];
- an **interpolation model** fills the gap in trajectories by linearly interpolating these detections based on the ones at the edges of the gaps.

While the *CP Associator* cannot be disabled (as it would prevent association), the other two modules are optional.

C. TrackletCutter - Cutting tracklets on overlapping sections

While the sensitivity to occlusions of most trackers often leads to fragmentation (i.e. tracks being cut into multiple tracklets), we developed a module designed to separate tracklets that are at risk of containing multiple different objects of interest. As we know that this risk is at its highest when multiple objects of interest cross paths, it was decided to proceed as follows : as shown in Figure 2, whenever in one frame two detections have an overlap (IOU) that reaches a fixed threshold T_{TC}, the tracklets to which they belong are cut at that frame.

D. Tracklet modeling

We define our set of tracklets as T and each tracklet $t \in T$ is, as shown in Figure 3, described by:

- a frame: f_S (resp. f_E) the frame in which the first (resp. last) detection of the tracklet appears;
- a bounding box: (x_S, y_S, w_S, h_S) (resp. (x_E, y_E, w_E, h_E)) the mean of the spatial coordinates of the six bounding boxes following the first one (resp. preceding the last one) of the tracklet;
- a speed: (v_S^x, v_S^y) (resp. (v_E^x, v_E^y)) the mean speed of the centers of the the six bounding boxes following the first one (resp. preceding the last one) of the tracklet.

If t is formed by fewer than ten detections, the averaging of the six first and last bounding boxes is not performed. The speed is then computed between the first (resp. last) two bounding boxes and the spatial coordinates are those of the first (resp. last) bounding box. However, doing this for each tracklet would have led us to put too much weight on the quality of these first and last bounding boxes that are by essence the least representative of the object of interest (as we can infer that the track has been separated at them because of some defect or occlusion), thus the choice of averaging the few following (resp. preceding) detections. The choice of working on six bounding boxes is however arbitrary and could be refined in future works.

Figure 2. Workings of the $TrackletCutter$ with $T_{TC} = 0.3$: Exploration of each frame one by one; as soon as the overlap of two bounding boxes reaches T_{TC} their tracklets are cut.

Figure 3. Construction of the two representative bounding boxes of a tracklet by averaging bounding boxes at its start and end.

E. Associating tracklets with Constraint Programming

Once the tracklets have been modeled, the goal is to associate them using our *CP Associator*. To do so, we need to model the problem following the CP paradigm, to define our use of constraints (both filtering and assigning marginals) and to define the methods we use to explore the solution space.

1) Modeling the association problem in a CP *paradigm:* A CP model is given by a finite set of variables, each taking its value from a finite set called its domain. Constraints are then specified on the combinations of values that these variables can take. This model defines a *Constraint Satisfaction Problem* (*CSP*). In our case, the tracklet association phase is modeled as:

- S our set of successor variables such that, for every tracklet i, $S(i)$ is its successor, meaning that $S(i)$ immediately follows i in the same trajectory.
- For every tracklet i, the domain $D(i)$ contains every tracklet that starts temporally after i ends, and a stopping value (meaning that i is the last tracklet of its trajectory).
- C is our set of constraints, which we use to filter the domains of each successor and to affect a score to each tracklet-successor pair.

2) Using constraints to filter: Constraints are most often used to restrict the domains of variables. We use them in that fashion to ensure the following character-

istics for our trajectories:

allDifferent: No detection should be found in multiple trajectories and therefore no tracklet should be assigned to multiple trajectories. To accomplish that goal, we used the *allDifferent* constraint. Applied to the whole set of variables, it ensures that no two variables are assigned the same value. It is given by:

$$\forall(i,j) \in T^2 \,|\, i \neq j, \ S(i) \neq S(j) \tag{1}$$

Temporal consistency: As we suppose in this part of our model that our tracklets are perfect (namely that each detection in a specific tracklet belongs to a single object), there should be no overlap in time between tracklets affected to a single trajectory. Therefore, we define the temporal consistency constraint as follows:

$$\forall(i,j) \in T^2, j \in D(i) \Rightarrow f_E(i) < f_S(j) \tag{2}$$

3) Score-based constraints: We also propose to use constraints in a different manner, that is to assign a score to each pair (t, s) (where t is a tracklet and $s \in D(t)$ is a successor) based on a given characteristic c. Each constraint is based on a distance $c(t, s)$ between t and s. These different kinds of distances (that can be, as we will see below, temporal or spatial for example) are then transformed into scores $S_c(t, s)$ ranging from 0 to 1 where:

- $S_c(t, s) = 0$ leads to the immediate removal of s from $D(t)$ the domain of the successors of t.
- $S_c(t, s) = 1$ leads to the immediate assignment of s to $S(t)$ i.e. s becomes the successor of t.
- $S_c(t, s_1) \geq S_c(t, s_2)$ means that according to characteristic c, s_1 is a more likely candidate than s_2 to be the successor of t.

Building scores: Each score-based constraint is assigned three thresholds that will help define their behavior:

- T_{50}^c: value of the distance for which we set the score at 50% (i.e. 0.5).

Figure 4. Representation of the computation of a constraints score : the thick black line represents S (gaussian of peak 1, mean 0 and standard deviation $\frac{1}{2ln(2)}T_{50}^c$), the red line represents the real score: bounded above by U and below by L, falls to 0 when reaching T_0^c. To enhance visibility we set U = 0.9 and L = 0.1. Typically we set $L = 10^{-6}$ and $U = 1 - L$.

Figure 5. Computation of the characteristics evaluated by the *predicted IOU* (*PIOU*) and *predicted center distance* (*PCD*) constraints : the ending bounding box of the tracklet is projected onto the first frame of the successors candidates. The starting bounding boxes of the successors are compared with the projection based on their overlap and the distance between their centers.

- T_{end}^c: value of the distance between the tracklet t and the fictional tracklet which represents the stopping of the trajectory at t. It is indeed essential to compare the set of possible successors to the possibility of associating with none of them (which would mean that t is that last tracklet of its trajectory).
- T_0^c: value of the distance beyond which the examined successor s is removed from $D(t)$. This threshold may or may not be activated, but if it is and it is reached for a given pair (t, s), $S_c(t, s)$ is then equal to 0 and s is removed from $D(t)$.

As shown in Figure 4, as long as $c(t, s) \leq T_0^c$ (where it automatically falls down to 0), the score of the association of t and s according to the characteristic c, $S_c(t, s)$, is calculated as follows. We compute

$$S = exp(\frac{-c(t,s)^2}{2\sigma_c^2}) \qquad (3)$$

where

$$\sigma_c = \frac{T_{50}^c}{2ln(2)} \qquad (4)$$

so that $S = 0.5$ for a distance of T_{50}^c. Finally the actual score is bounded as follows

$$S_c(t, s) = max(L, min(S, U)) \qquad (5)$$

where L is the lower bound of the score and U the upper bound (so that it does not reach 0 or 1).

Constraint on time spacing: The first kind of score-based constraint we developed favors a small temporal distance between a tracklet and its successor. For each pair (t, s), we get the metric $td(t, s)$ (where td stands for *time distance*) such that:

$$td(t, s) = f_S(s) - f_E(t) \qquad (6)$$

Constraints on dynamics: We also decided to built constraints that aim to maintain the trajectories as smooth as possible by minimizing the discontinuities in acceleration. For each pair (t, s) we then obtain the metrics $ad(t, s)$ and $sd(t, s)$ (for *angle difference* and *speed norm difference*) such that:

$$ad(t, s) = (\widehat{\overrightarrow{v_E(t)}, \overrightarrow{v_S(s)}}) \qquad (7)$$

$$sd(t, s) = \left|\overrightarrow{v_S(s)}\right| - \left|\overrightarrow{v_E(t)}\right| \qquad (8)$$

Constraints on a predicted position: We can suppose that with the help of the aforementioned *time distance* constraint that the tracklets most likely to be associated are those that are not too temporally distant. As the shorter the time interval, the less the object of interest can change its speed, direction and even position in the image, we decided to add a constraint on a predicted position. Following the example of a *Kalman filter* [4], we consider that projecting a bounding box using its speed onto subsequent frames not too far apart is a relatively efficient prediction mechanism. Therefore we propose two constraints, which are described in Figure 5 : they compare the predicted position of the considered object and the evaluated successor by *IOU* or by the distance between their centers (which is largely considered to be more robust to occlusions [14]).

4) Adaptation to video sequences properties: As the video sequences we work with have different characteristics regarding dimensions and capture speed (measured in frames per second or *FPS*), we chose to adapt the parameters of the score-based constraints. Indeed, affecting a *time distance* score of 0.5 to a pair of tracklets separated by six frames has a very different meaning for a sequence of 12 FPS compared to one four times faster.

Therefore we decided to adapt the *time distance* constraint to the FPS of the sequence, the *predicted center distance* constraint to the size of the image (represented by proxy by the length of the diagonal of the image), and the *speed norm difference* constraint to the FPS and the diagonal.

5) Marginals: Once each of the successor domains $D(t)$ have been restricted by the activated relevant constraints, the remaining (t, s) pairs (where $t \in T$ and $s \in D(t)$) get a score from each activated score-based constraint (as explained before). That leaves us with up to 5 scores per pair that we would like to use to guide our solution exploration. To do so, we combine these into marginals, following in a way the example set by [21] where the authors try to minimize a sum of energies. We compute these marginals $M(t, s)$ as the product of scores normalized over $D(t)$:

$$M(t, s) = \frac{\prod\limits_{c} S_c(t, s)}{\sum\limits_{k \in D(t)} \prod\limits_{c} S_c(t, k)} \qquad (9)$$

Usually when Belief Propagation (BP) is used in CP, the marginals built represent the density of solutions resulting from the branching of a constraint for the considered variable-value pair. It is used here to convey our marginals.

6) Exploration method: A branching heuristic called *MaxMarginal* has been developed in *MiniCPBP* [20]. It consists in guiding the construction of the search tree exploring the pairs tracklet-successor by descending order of marginal. Regarding the exploration strategy, to prioritize staying close to the model by promoting high marginals association first, we use *Depth First Search* (DFS) that consists in exploring the search tree by taking deviations as low as possible in the search tree if a valid is not found at first.

F. Interpolation model

Once the tracklets are associated with each other, it is likely that the resulting trajectories will have gaps, i.e. sequences of frames in which the object disappears before reappearing. This is why we decided to integrate a simple interpolation model in our method, which works as follows:

As shown in Figure 6, we identify the holes in each trajectory and if they are smaller than a threshold, $maxGapSize$, we fill them by making a linear interpolation from the detection preceding the hole to the one following it. Simply put, we consider that the object has moved (and changed shape or size) at a constant speed from the detection that precedes the hole to the one that follows it, and we add all the missing detections to the trajectory.

Figure 6. Workings of the interpolation model for $maxGapSize = 4$: two detections are being placed to fill a hole, by placing centers regularly between the edges of the hole and then interpolating their dimensions linearly.

IV. EXPERIMENTS AND RESULTS

A. Evaluation Dataset

We have chosen to evaluate our model on the training set of the *MOT17* [22] challenge based on the detections of the best proposed detector, *FRCNN* [2]. This dataset represents a reference in the field and presents many difficulties that are particularly interesting to confront. Whether it is the often high occlusion rates, turbulence, moving cameras, strong variations in light exposure, sometimes subjective POV and other times very elevated POV, or the numerous reflections present in these videos, we are dealing with extremely varied situations that should allow us to evaluate our model in the majority of situations that can happen in urban settings.

B. Evaluation metrics

We evaluate our method using three of the main *MOT* metrics: *MOTA* (and *MOTP*) [23] that mainly measure the quality of detections, and are used very broadly in the literature, *IDF1* [24] that refers more to the quality of the association between detections, also widely used in the literature, and *HOTA* [25] a more recent metric that accounts for both the performance in terms of association of detections and the quality of these detections.

C. Parameters

We tested multiple combinations of parameters for each of our modules (tried to activate or not each of the constraints) to find interactions and select the best configuration. Table I represents the best configuration

Table I
FINAL MODEL CONFIGURATIONS USED IN ABLATION TESTING

Module		Best configuration
Association	TimeDistance	$td_{50} = 1$
		$td_{end} = 3$
	PredictedCenterDistance	$pcd_{50} = 0.02$
		$pcd_{end} = 2$
	PredictedIOU	$piou_{50} = 0.75$
		$piou_{end} = 2$
	AngleDifference	$disabled$
	SpeedNormDifference	$disabled$
TrackletCutter		$T_{TC} = 0.5$
Interpolation		$maxGapSize = 42$

Table II
RESULTS OF THE DIFFERENT MODEL COMPONENTS IN ABLATION ON THREE DIFFERENT TRACKERS APPLIED TO THE *MOT17* TRAINING SEQUENCES. TC: *TrackletCutter*, CP: *CP Associator*, INT: *INTERPOLATION MODEL*. FOR ALL METRICS, HIGHER IS BETTER.

Method	HOTA	MOTA	IDF1
IOU-Tracker	43.04%	49.74%	50.27%
IOU-Tracker + CP	45.34%	49.91%	53.78%
IOU-Tracker + TC + CP	45.47%	49.90%	54.11%
IOU-Tracker + CP + Int	46.04%	**50.62%**	54.38%
IOU-Tracker + TC + CP + Int	**46.18%**	50.49%	**54.74%**
SORT	42.80%	48.54%	50.63%
SORT + CP	45.40%	48.71%	54.34%
SORT + TC + CP	45.14%	48.70%	53.96%
SORT + CP + Int	**46.35%**	**49.46%**	**54.92%**
SORT + TC + CP + Int	46.06%	49.39%	54.58%
Tracktor	55.18%	61.81%	65.06%
Tracktor + CP	56.39%	61.87%	67.40%
Tracktor + TC + CP	55.89%	61.85%	66.77%
Tracktor + CP + Int	**56.98%**	**62.99%**	**67.88%**
Tracktor + TC + CP + Int	56.49%	62.96%	67.29%

we found by applying our model to the *MOT17* training data. We found during these calibration sessions that the vast majority of high scoring configurations were those that did not give the ability to filter successors (i.e. reduce their score to 0) to score-based constraints. We therefore disabled this ability of constraints that only guide the search in the configuration presented below.

Concerning the exploration of the solutions, it turned out that by pushing this one even up to the $10,000^{th}$ valid solution explored for multiple configurations of parameters and constraints, we did not obtain better results than by stopping at the first one found by branching on the maximal marginals (except for the very bad models, which obtained in all cases worst solutions than the initial tracker), so we decided to stop at the first valid solution in our exploration. The code for our method can be found at github.com/reminahon/tracklet_associator.

D. Results and Discussion

Results are given in Table II. It can be noted that whatever the tracker we apply it to, our model allows to obtain improvements of several percents on the three scores that interest us: *HOTA*, *MOTA* and *IDF1*. Concerning the *HOTA*, the main metric of our evaluation,

we obtain an improvement of 3.14% for *IOU-Tracker*, 3.55% for *SORT* and 1.8% for *Tracktor* which already had a rather high score (13% more than the two others originally) which shows that our module is likely effective on any type of tracker independently of their initial level of performance or their tracking paradigm.

Our goal was mainly to improve data association with our *CP Associator*, but we also addressed the detection phase with the *interpolation* module. *HOTA* and *IDF1* are the two metrics that are the most sensitive to the quality of the associations, as opposed to *MOTA* that is not very sensitive to the data association quality, but more sensitive to the detections quality. We observe that the *CP* association model is the module that allows the most improvements in terms of *HOTA* and *IDF1* (70% of the improvements on average) to the results of the three trackers. The rest of the improvements are mainly brought by the interpolation model which allows to improve the MOTA by more than one point for *Tracktor*, in particular, by adding missing detections.

However, the *IOU-Tracker* is the only tracker for which the *TrackletCutter* really allows an improvement of the results. This may be due to the fact that this tracker has more errors due to occlusions detected by our method. Still, it seems that our model performs adequately without the *TrackletCutter*. It turns out that this is the part of the model that requires the most computation time, for little to no improvement. We would therefore advise not to use it for any other tracker than *IOU-Tracker*. Moreover, on the whole MOT17 training set, applying our model to get the improved trajectories takes between 30 to 60 seconds without the *TrackletCutter* and up to 5 minutes with it. Even without the use of the *TrackletCutter*, our model retains interest insofar as trackers tend to suffer from fragmentation which we correct by our association. One could even postulate that the more a tracker suffers from fragmentation, the better our post-processing can help its tracking performance.

V. CONCLUSION

We presented a method that can be used as a post-processing step for any state-of-the-art multi-object tracker to improve its association performance as we have been able to show by testing it on the trackers *IOU-Tracker*, *SORT* and *Tracktor* on the *MOT17* dataset. This demonstrates its competitiveness in the field of pedestrian tracking. In addition we propose here the first association model based on Constraint Programming with Belief Propagation. Furthermore, a strength of our method for future improvements relies on our modularity: each module we propose (whether it is the *TrackletCutter*, the association model or the interpolation one) can be substituted with another one that would accomplish the same function. New constraints based on

other characteristics (such as appearance for example) can also be added without any major changes in the architecture of the model.

REFERENCES

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[2] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[3] E. Bochinski, V. Eiselein, and T. Sikora, "High-speed tracking-by-detection without using image information," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017, pp. 1–6.

[4] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 03 1960.

[5] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple Online and Realtime Tracking," *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3464–3468, Sep. 2016, arXiv: 1602.00763.

[6] G. D. Evangelidis and E. Z. Psarakis, "Parametric image alignment using enhanced correlation coefficient maximization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 10, pp. 1858–1865, 2008.

[7] C. Novak and S. Shafer, "Anatomy of a color histogram," in *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1992, pp. 599–605.

[8] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, 2005.

[9] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, p. 583–596, Mar 2015.

[10] S. Sun, N. Akhtar, H. Song, A. Mian, and M. Shah, "Deep Affinity Network for Multiple Object Tracking," *arXiv:1810.11780 [cs]*, Jul. 2019, arXiv: 1810.11780.

[11] Y. Xu, Y. Ban, G. Delorme, C. Gan, D. Rus, and X. Alameda-Pineda, "TransCenter: Transformers with Dense Queries for Multiple-Object Tracking," *arXiv:2103.15145 [cs]*, Mar. 2021, arXiv: 2103.15145.

[12] A. Girbau, X. Giró-i-Nieto, I. Rius, and F. Marqués, "Multiple object tracking with mixture density networks for trajectory estimation," *CoRR*, vol. abs/2106.10950, 2021. [Online]. Available: https://arxiv.org/abs/2106.10950

[13] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, "Tracking without bells and whistles," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 941–951, Oct. 2019, arXiv: 1903.05625.

[14] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking Objects as Points," *arXiv:2004.01177 [cs]*, Aug. 2020, arXiv: 2004.01177.

[15] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes, "Globally-optimal greedy algorithms for tracking a variable number of objects," in *CVPR 2011*. Colorado Springs, CO, USA: IEEE, Jun. 2011, pp. 1201–1208.

[16] D. Stadler and J. Beyerer, "Improving multiple pedestrian tracking by track management and occlusion handling," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 10 953–10 962.

[17] L. Wen, W. Li, J. Yan, Z. Lei, D. Yi, and S. Z. Li, "Multiple Target Tracking Based on Undirected Hierarchical Relation Hypergraph," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 1282–1289, iSSN: 1063-6919.

[18] A. Pineault, G. Bilodeau, and G. Pesant, "Tracking road users using constraint programming," *CoRR*, vol. abs/2003.04468, 2020. [Online]. Available: https://arxiv.org/abs/2003.04468

[19] G. Pesant, "A constraint programming primer," *EURO Journal on Computational Optimization*, vol. 2, no. 3, pp. 89–97, Aug. 2014.

[20] ——, "From Support Propagation to Belief Propagation in Constraint Programming (Extended Abstract)," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, Jul. 2020, pp. 5100–5104.

[21] A. Milan, S. Roth, and K. Schindler, "Continuous Energy Minimization for Multitarget Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 1, pp. 58–72, Jan. 2014.

[22] A. Milan, L. Leal-Taixé, I. D. Reid, S. Roth, and K. Schindler, "MOT16: A benchmark for multi-object tracking," 2016. [Online]. Available: http://arxiv.org/abs/1603.00831

[23] K. Bernardin and R. Stiefelhagen, "Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, no. 1, p. 246309, 2008.

[24] E. Ristani, F. Solera, R. S. Zou, R. Cucchiara, and C. Tomasi, "Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking," *arXiv:1609.01775 [cs]*, Sep. 2016, arXiv: 1609.01775.

[25] J. Luiten, A. Osep, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixé, and B. Leibe, "HOTA: A Higher Order Metric for Evaluating Multi-object Tracking," *International Journal of Computer Vision*, vol. 129, no. 2, pp. 548–578, 2021.