Tracking Pedestrians using Constraint Programming[*]

Alexandre Pineault, Guillaume-Alexandre Bilodeau, and Gilles Pesant

Polytechnique Montréal, Montréal, Canada
alexandre.pineault@polymtl.ca, gabilodeau@polymtl.ca,
gilles.pesant@polymtl.ca

Abstract. This paper presents a constraint programming (CP) approach for the data association phase found in the tracking-by-detection paradigm of the multiple object tracking (MOT) problem. Such an approach can solve the data association problem more efficiently than graph-based methods and can handle better the combinatorial explosion occurring when multiple frames are analyzed. Because our focus is on the data association problem, our MOT method only uses simple image features, which are the center position and color of detections for each frame. Constraints are defined on these two features and on the general MOT problem. Two filtering layers are used in order to eliminate detection candidates before using CP and to remove dummy trajectories produced by the CP solver. Our proposed method was tested on a pedestrian tracking dataset and outperforms several classic data association methods such as minimum-cost flow.

Keywords: Multiple object tracking · Data association · constraint programming.

1 Introduction

A very common approach to tackle the multiple object tracking (MOT) problem uses the tracking-by-detection paradigm, which divides the task into two smaller ones: the detection of objects in the video frames and the association of these detections between frames to form trajectories for the objects of interest [9, 11, 1, 7]. These trajectories can represent data about road users that is useful to solve higher-level problems such as improving security in our streets. Tracking objects can help design more secure roads by analyzing road user behaviour and finding anomalies in their trajectories before incidents actually happen [12].

This work focuses on the data association step of MOT methods. We assume that our detections come from an off-the-shelf object detector (Figure 1). The main challenges of data association are: 1) handling occlusions which occur when a tracked entity is hidden completely or partially by other objects such as other pedestrians or untracked foreground elements, 2) managing incoming

[*] This research was supported by an IVADO fundamental research grant.

and outgoing objects of interest, which means that a trajectory may not last the whole video sequence, and 3) taking into account the imperfect performance of the detector (bounding boxes that do not bound perfectly an object, missing objects, false detections).

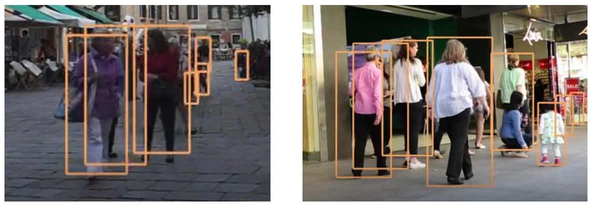

Fig. 1. Examples of detections (showed as orange bounding boxes) around objects of interest. As they come from an imperfect detector, mistakes are often present.

The data association step can be viewed as a combinatorial problem between current tracks (a track is a trajectory being build by the tracker) and new detections. For this reason, the use of constraint programming (CP) is an interesting option to explore since CP has long been used to solve various combinatorial problems [15].

This paper presents a MOT method that produces trajectories by matching recurring objects together using a set of constraints that use the center position and the color of detections. Our main contribution is a CP model that is adapted to the MOT problem. Some filters are defined before and after the CP solving phase. Our method is compared with others based on data association strategies and with similar simple features as the ones we are using. We show that in the case of simple features, our CP-based data association method gives better results than min-cost flow. Using additional constraints, the proposed CP-based data association can be extended to include other features.

2 Background and Related Work

State-of-the-art performance in multiple object tracking is currently obtained with methods using deep neural networks to extract many automatically-learned features [4, 11]. One of these methods [4] is using a convolutional neural network (CNN) to build two classifiers: one for separating object categories and another for instance classification which will differentiate objects of a same category. In this approach, the previous classifiers are connected to a particle filter which, combined with the appearance model, makes a prediction of where the object will appear next. The idea of using object categories in tracking was also exploited in the work of Ooi et al. [6]. These state-of-the-art methods focus mainly in correcting missing and spurious detections using a prediction method and defining robust features for data association. They do not study how to make the data association itself. They show that a strong appearance model and predicting where an object should appear next to filter object detections are two important components of a MOT system. However, they use a limited data association strategy, as data association decisions are not taken over several frames by combining several observations of the objects in a video. Occlusions are known to be better resolved over several frames. Therefore, a more robust data association method is always desirable.

Typical data association methods used in tracking are the Hungarian algorithm, the joint probabilistic data-association filter (JPDAF), and the minimum-cost flow algorithm. The Hungarian algorithm finds a maximum cost matching in a bipartite graph where the costs are on edges [14]. As the assignment problem can be formulated as a bipartite graph, this algorithm is one way to get the solution. It was used in several works. For example, the Hungarian algorithm can be combined with tracklets (incomplete track parts) to make the associations [1]. These tracklets are ranked according to a confidence metric considering occlusions, number of frames covered and how well the detections fit. Online detections are matched to existing tracklets with high confidence using the Hungarian algorithm maximizing the affinity level. The next step is to globally match these new associations with low level detections using once again the Hungarian algorithm.

JPDAF is a statistical method that can track objects based on what is the most likely outcome for each trajectory. It considers any detection available, but also the possibility of a missing object or a false detection. It was used several times for the MOT problem, for example, in the work of [8] and [2].

Lastly the min-cost flow algorithm combines a cost function with a greedy algorithm in order to obtain the required tracking associations [7]. The goal here is to formulate the data association as a minimum-cost flow problem, then to compute shortest paths on the flow network with detections at each frame as nodes, from the first appearance of an object to its last appearance in the scene. This process helps ensure that the resulting trajectories are as smooth as possible.

3 Methodology

Our tracking approach can be summarized as follows. First, detections are obtained for every video frame using a pedestrian detector. These detections are then filtered based on their confidence. Then, the detections are used to instantiate the CP model and the model is solved by forming tracks with the detections. Finally, some dummy tracks are removed before outputting the final result (a set of trajectories). Our method is detailed in the following.

3.1 Constraint Programming Model

Main Variables In order to associate tracks to detections, variables

$$t_{ij} \in \{1, 2, \ldots, \tau\} \qquad \forall\, 1 \leq i \leq m,\ 1 \leq j \leq n_i \tag{1}$$

identify to which track (out of τ available ones) detection j at frame i belongs, where m is the number of frames and n_i the number of detections at frame i.

Such a representation is convenient to avoid unnecessary symmetries since it uses the smallest number of possible variables: one per detection. However, referring to consecutive detections of a given track is not easy because there is no way to know the indices of the variables that will be part of a same track before the solving step. Since we need this to express some of our constraints, a second array of variables is defined,

$$d_{ik} \in \{1, 2, \ldots, \tau\} \qquad \forall\, 1 \leq i \leq m,\ 1 \leq k \leq \tau \tag{2}$$

identifying the detection assigned to track k at frame i. Whenever $\tau > n_i$ for some frame i, values greater than n_i do not represent actual detections — they are however necessary because we will require that detections be uniquely associated to tracks (see Equations 4 and 5).

To ensure that these two dual representations remain consistent with each other, we link them using an `inverse` constraint

$$d_{i\star} = \texttt{inverse}(t_{i\star}) \qquad \forall\, 1 \leq i \leq m \tag{3}$$

relating each variable to its counterpart in the other array.

Frame Consistency The next step is to ensure that it is impossible to find a specific track or detection more than once at a given frame. This is a common requirement for which CP provides an `alldifferent` constraint. Such constraints have been specified for this purpose using the same variables as the previous equations.

$$\texttt{alldifferent}(t_{i\star}) \qquad \forall\, 1 \leq i \leq m \tag{4}$$
$$\texttt{alldifferent}(d_{i\star}) \qquad \forall\, 1 \leq i \leq m \tag{5}$$

Position Constraints After the model structure is set, constraints on features such as position can be considered. First, an array of integers is created for each dimension in a frame (x and y coordinates). These arrays, indexed by frame and then by detection ID, contain the center of each detection bounding box. It is then possible to formulate a constraint restricting the distance between two consecutive detections in a track by stating that the one-dimensional distance along the X and the Y axes separating them may not be greater than thresholds λ_x and λ_y :

$$|x_{i,d_{ik}} - x_{i+1,d_{i+1,k}}| \leq \lambda_x \quad \forall\, 1 \leq i < m, 1 \leq k \leq \tau \tag{6}$$
$$|y_{i,d_{ik}} - y_{i+1,d_{i+1,k}}| \leq \lambda_y \quad \forall\, 1 \leq i < m, 1 \leq k \leq \tau \tag{7}$$

The same process has been tested using object scale indicators but this feature did not help to improve the solution.

Appearance Model An appearance model is useful for minimizing the risk of mismatch between two nearby objects. Our proposed appearance model assigns a color class label to each detection available. The color classes are obtained by clustering observed object color histograms in several videos. Clustering is performed with K-Means using the Bhattacharya distance between the color histograms. The learned classes are then used during tracking. A detection in a frame is assigned the color class label of the nearest cluster.

A regular constraint is applied to each track preventing the association of two objects with contrasting colors. Possible transitions between color classes for a given object at each frame is governed by a state machine. Transitions between similar color classes are allowed but a cost is applied based on the Bhattacharya distance between the class centers. The solver will use the sum of all these costs as a minimization objective.

$$c_{ik} = \texttt{colour}(d_{ik}) \tag{8}$$

$$\texttt{costregular}(c_{\star k}, \mathcal{A}, C, a_k) \qquad \forall\, 1 \leq k \leq \tau \tag{9}$$

$$\texttt{min} \left(\sum_{k=1}^{\tau} a_k \right) \tag{10}$$

In this equation, c represents the list of color class labels for the track detections acting as transition variables between the state machine different states. $\mathcal{A}$ and C correspond respectively to the authorized transitions and the cost matrix for every possible transition. Variable a_k represents Track k's total cost computed by the regular constraint.

Each color class is represented by two states, depending on whether the object is currently being occluded or not (e.g. red object and red occluded object). Using different states for occlusions allows to set a cost for transitions from a valid detection to an absence of detection while preserving the object appearance even when it is occluded for an extended duration (see Figure 2).

3.2 Solving Strategy

Variable Selection As stated in the introduction, CP explores the combinatorial space of solutions by branching in a search tree that enumerates all possible solutions. To solve our model, we first branch on the t_{ij} variables since it is the main array containing the smallest number of variables necessary to specify a solution whereas in the case of the d_{ij} variables, all possible tracks are instantiated in case they are needed. To allow the state machine to handle occlusions or the absence of a detection for a given track, it is necessary to branch on the c_{ij} variables. The lexicographic order specified will make sure that branching will always at first be made on t_{ij} variables, then on c_{ij} variables. In both cases, variables are considered frame by frame, then detection by detection in ascending order.

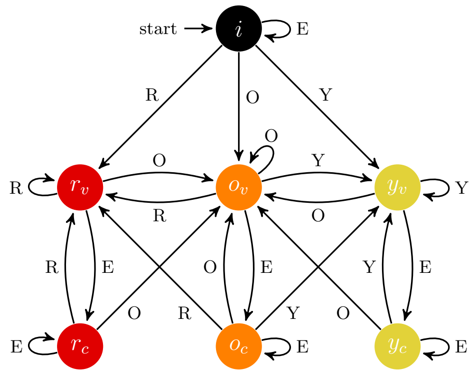

Fig. 2. Small state machine example illustrating the doubled states: visible (v) and occlusion (c) for three color classes: red (r), orange (o) and yellow (y). Transition values correspond to the detection color (R, O, Y) or empty (E) when there is no detection.

Value Selection Since the t_{ij} variables are fixed from the first to the last frame, the use of prediction becomes an interesting option to guide the search. By computing every distance between possible pairs of consecutive detections, it is possible to obtain a ranked list that the solver can use to branch on closest detections first for each t_{ij} excepted for the last frame. This way, we minimize the distance between each consecutive pair of objects without altering the model objective oriented towards the appearance preservation.

The inherent symmetries are addressed in the branching to control the number of opened tracks available as candidate for t_{ij} variables. The domain of each of these variables is reduced to all previously used tracked index plus one other that may be opened if necessary. This strategy was applied before to the Steel Mill Slab Design Problem [13].

3.3 Post Solving Computations

Once the solver has completed the associations, another type of filtering is required due to how the model is built. The model is unable to eliminate detections entirely — the only choice that it has is to put each of them in a dummy track, which will have to be deleted afterward. To delete them, a filter removes every track that contains fewer than β_D detections.

After the suppression of the unwanted tracks, it is possible to fill gaps in the remaining tracks. A small state machine searches for places where there are γ_D or less consecutive missing detections and fills these empty spots by adding new detections using linear interpolation to find the correct center position and size of the bounding box for each one. This has to be done carefully because a too large gap probably means that an identity change has happened. Therefore, filling missing detections in these situations would probably mean that false detections are added.

4 Preliminary Experiments

We tested our method on the 2D MOT 2015 challenge dataset [5]. The goal of this challenge is to track pedestrians in eleven video sequences. The CP solver used for these experiments is IBM ILOG CP version 1.6.

4.1 Evaluation metrics

To evaluate our method, we use the standard MOT metrics that account for three types of errors at each frame i: 1) Identity switches (IDS_i) or the number of mismatches, 2) the missing detections or false negatives (FN_i) which are the number of objects that are not tracked, and 3) the false positives (FP_i) which stands for the number of detections representing no object of interest. With these values it is possible to compute the *MOTA* score [3]

$$MOTA = 1 - \frac{\sum_i (FP_i + FN_i + IDS_i)}{\sum_i N_i} \tag{11}$$

where N_i is the total number of ground-truth objects per frame and is summed throughout the entire video.

A second metric known as *IDF1* balances the recall and precision for each trajectory into a single value. This is done by comparing the total number of accurately matched detections to the average number of ground truth and given detections [10]. Finally mostly tracked trajectories (MT) is the ratio of ground truth that possess a matching hypothesis for at least 80% of their life span and mostly lost trajectories (ML) is the number of times this ratio is under 20%. Fragmentation $Frag$ indicates how many times any trajectory got interrupted over its length.

4.2 Results

Table 1 compares our method with other approaches. The first two rows, *HWDPL* [4] and *AMIR15* [11], are to show the current performance of state-of-the-art methods that uses multiple learned features, learned cost function and prediction in future frames. The last four entries includes our method and three classic approaches using similar simple features as ours: *DP NMS* (Min-cost flow) [7], *TC ODAL* (Hungarian algorithm) [1] and *JPDA OP*, a JPDAF implementation from the MOT challenge website [1].

Table 1 contains the cumulative scores for the eleven video sequences of the challenge. This implies that our method was used on eleven different instances to produce the required tracks. These results show that there are still many improvements to be made before we can match the performance of the state-of-the-art approaches, but it it also shows that our method surpasses many classical approaches using similar features. The overall process took 91 seconds to complete for 5783 frames (607 seconds).

[1] https://motchallenge.net

Method	*MOTA*	*IDF*1	*MT*	*ML*	*FP*	*FN*	*IDS*	*Frag*
HWDPL [4]	38.5	47.1	8.7%	37.4%	4005	33203	586	1263
AMIR15 [11]	37.6	46.0	15.8%	26.8%	7933	29397	1026	2024
DP NMS [7]	14.5	19.7	6.0%	40.8%	13171	34814	4537	3090
TC ODAL [1]	15.1	0.0	3.2%	55.8%	12970	38538	637	1716
JPDA OP	3.6	7.5	0.4%	96.1%	1024	58189	29	119
CP (Ours)	17.0	13.3	2.5%	58.4%	4872	43170	2973	3077

Table 1. Tracking results on the 2D MOT 2015 benchmark.

4.3 Main Challenges

Scalability MOT is a problem where the computation time is limited since completing the tracking in real time is desired in many situations. Our approach using CP is not currently close to this level of efficiency. Tracking video sequences contains often more than a thousand frames which increase the number of possible combinations before obtaining a solution. The total number of objects throughout the complete sequence is also impacting the computation since it directly affects the domain of the main branching variables. As a solution to this, the data association could be applied to blocks of consecutive frames (say 100 frames). Recall that usually data association is performed only on two consecutive frames. Considering more frames allows taking better long-term decisions. It is therefore not mandatory to perform the data association with all the frames. Finally, the complexity of the appearance model (i.e. the number of possible appearance values) affects directly the time required to test each variable-value combination.

Appearance Model The appearance model is important to indicate the presence of occlusions. The precision with which we describe detections will impact the tracking accuracy. One approach may be to use vectors to describe the object (histogram of oriented gradients, deep features). However, they are difficult to include in a CP model without doing clustering first which may reduce the precision of object descriptions. As said before, a complex model will also slow down the resolution process.

5 Conclusion

This paper introduced a novel way to make data association using constraint programming. More precisely the center position of objects and their colors are used as main features to investigate if CP can be a valid approach to solve this problem. There are still improvements required to be able to solve large instances and obtain more competitive results. Future work will include developing a more complex model that considers a larger number of features and optimizing the search strategy in order to improve run-time.

References

1. Bae, S., Yoon, K.: Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1218–1225 (June 2014). https://doi.org/10.1109/CVPR.2014.159
2. Bar-Shalom, Y., Daum, F., Huang, J.: The probabilistic data association filter. IEEE Control Systems Magazine **29**(6), 82–100 (Dec 2009). https://doi.org/10.1109/MCS.2009.934469
3. Bernardin, K., Stiefelhagen, R.: Evaluating multiple object tracking performance: The clear mot metrics. EURASIP Journal on Image and Video Processing **2008**(1), 246309 (May 2008). https://doi.org/10.1155/2008/246309, https://doi.org/10.1155/2008/246309
4. Chen, L., Ai, H., Shang, C., Zhuang, Z., Bai, B.: Online multi-object tracking with convolutional neural networks. In: 2017 IEEE International Conference on Image Processing (ICIP). pp. 645–649 (Sep 2017). https://doi.org/10.1109/ICIP.2017.8296360
5. Leal-Taixé, L., Milan, A., Reid, I., Roth, S., Schindler, K.: MOTChallenge 2015: Towards a benchmark for multi-target tracking. arXiv:1504.01942 [cs] (Apr 2015), http://arxiv.org/abs/1504.01942, arXiv: 1504.01942
6. Ooi, H., Bilodeau, G., Saunier, N., Beaupré, D.: Multiple object tracking in urban traffic scenes with a multiclass object detector. In: ISVC. Lecture Notes in Computer Science, vol. 11241, pp. 727–736. Springer (2018)
7. Pirsiavash, H., Ramanan, D., Fowlkes, C.C.: Globally-optimal greedy algorithms for tracking a variable number of objects. In: CVPR 2011. pp. 1201–1208 (June 2011). https://doi.org/10.1109/CVPR.2011.5995604
8. Rezatofighi, S.H., Milan, A., Zhang, Z., Shi, Q., Dick, A., Reid, I.: Joint probabilistic data association revisited. In: 2015 IEEE International Conference on Computer Vision (ICCV). pp. 3047–3055 (Dec 2015). https://doi.org/10.1109/ICCV.2015.349
9. Riahi, D., Bilodeau, G.A.: Online multi-object tracking by detection based on generative appearance models. Computer Vision and Image Understanding **152**, 88 – 102 (2016). https://doi.org/https://doi.org/10.1016/j.cviu.2016.07.012
10. Ristani, E., Solera, F., Zou, R.S., Cucchiara, R., Tomasi, C.: Performance measures and a data set for multi-target, multi-camera tracking. In: Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II. pp. 17–35 (2016). https://doi.org/10.1007/978-3-319-48881-3_2
11. Sadeghian, A., Alahi, A., Savarese, S.: Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 300–311 (Oct 2017). https://doi.org/10.1109/ICCV.2017.41
12. Saunier, N., Sayed, T., Ismail, K.: Large-scale automated analysis of vehicle interactions and collisions. Transportation Research Record **2147**(1), 42–50 (2010). https://doi.org/10.3141/2147-06, https://doi.org/10.3141/2147-06
13. Van Hentenryck, P., Michel, L.: The steel mill slab design problem revisited. In: Perron, L., Trick, M.A. (eds.) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. pp. 377–381. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)

14. W. Kuhn, H.: The hungarian method for the assignment problem. Naval Res. Logist. Quart. **2**, 83 – 97 (03 1955). https://doi.org/10.1002/nav.3800020109
15. Wallace, M.: Practical applications of constraint programming. Constraints **1**(1/2), 139–168 (1996). https://doi.org/10.1007/BF00143881