TRACKING IN STREAMED VIDEO BY UPDATING GLOBALLY OPTIMAL MATCHINGS

João F. Henriques, Rui Caseiro and Jorge Batista

Institute of Systems and Robotics, University of Coimbra

ABSTRACT

Matching methods such as the Hungarian algorithm have recently made an appearance as an alternative to classical tracking algorithms in computer vision, since they are able to find the set of tracks that optimizes well-defined criteria over a given video sequence. However, despite being globally optimal, they carry a cost: since they require complete knowledge of the sequence, such methods cannot work with continuous video streams, a crucial requirement of realistic video surveillance applications. We were able to use the recently proposed Dynamic Hungarian Algorithm in an innovative way, adapting it to the well-known sliding window methodology. The algorithm is able to run in real-time, while retaining its optimality. We tested our implementation on several datasets, tracking humans and vehicles, and obtained reliable results using the same set of parameters on all sequences.

Index Terms— Video surveillance, tracking, real-time, Dynamic Hungarian Algorithm

1. INTRODUCTION

Tracking is one of the most deceptively difficult problems in computer vision. This fact is especially jarring when we consider the effortlessness with which we humans are able to do the same task. The interest in this problem stems from its direct applications, such as video surveillance and robot navigation, but also because the most useful scene understanding tasks have complete, unambiguous object trajectories as a prerequisite.

Recent years have seen many developments in this area. Slowly, the shortcomings of previous methods are being dealt with, as computational resources allow the use of much more complicated algorithms. Knowledge of this evolution is necessary to understand how the work presented in this paper bridges the gap between two very different approaches to tracking.

The dominant approach for many years has been based on *object following*. Mimicking the inherently causal process that represents the movement of objects, these methods keep a state that summarizes previous observations and update it recursively with new observations. Examples are Kalman and particle filters [1], coupled with many other modules, such as track initializers, split and merge managers, and different forms of scene reasoning to deal with occlusion [2, 3]. However, notice that they generally assume that, given enough information about the past, optimal committing decisions can be made *now*. The decision-making process is local in time.

An attempt to delay decisions and explore different possibilities was made with Multiple Hypothesis Tracking (MHT) [4], which sparked many variants and is still widely used. It's not hard to imagine, though, the combinatorial explosion of considering all the different possibilities in a linear manner. This lead to the question of whether it is possible to explore the hypothesis-space more efficiently. It turns out it is possible, by resorting to the relatively new paradigm of globally optimal *object matching* [5, 6]. Given association scores for all possible pairs of detections over time (in this context, a detection is the occurrence of an object within a given frame), classical methods such as the Hungarian algorithm output the set of pairs that minimizes the sum of the scores, the *optimal matching*. Matchings have the property of no-overlap: a detection is matched to only one other detection. Taken in sequence they represent tracks, and so the optimization effectively operates on the hypothesis-space of all object tracks. Its worst-case time complexity is cubic in the number of detections.

But their effectiveness comes at a cost. Globally optimal methods, by their very definition, need access to all the detections at once. They work on videos of a few seconds or minutes, but are impossible to employ for continuous streams of video, potentially of many hours or days in the case of visual surveillance. They also require that the whole video is captured before any processing can be initiated. Such restrictions clearly rule out any sort of real-time response to the tracking results, crucial in practical applications.

Over this work, we set out to eliminate these restrictions. With only very mild and well-understood assumptions, our formulation uses a sliding window updated with the Dynamic Hungarian Algorithm, which doesn't compromise the optimality of the method.

2. COMBINATORIAL OPTIMIZATION FRAMEWORK

2.1. Simple matching

The basic framework takes the total set of detections of the whole video, $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$, and a cost matrix $C_{n\times n} = [c_{ij}]$, where c_{ij} is the cost of associating r_i to r_j. These costs can be obtained by any combination of measures between detections, such as color histogram correlation, position and size difference, or probabilistic models (as shown in Section 2.2). Impossible matches are expressed by infinite costs. This is important, to make sure that only matches going forward in time are allowed; formally, $t_i \geq t_j \Rightarrow c_{ij} = \infty$, with t_k the time instant of r_k. Notice that this already models occlusions, since a detection can be matched to another a few frames later if it leads to an optimal solution.

Given the matrix C, the Hungarian algorithm (also referred to as the Munkres algorithm) computes the optimal matching $S^*_{n\times n} = [s_{ij}]$, where $s_{ij} = 1$ if r_i is matched to r_j and 0 otherwise. This algorithm is referred many times in the literature [7] and there are numerous implementations available. It minimizes $\sum_i \sum_j c_{ij} s_{ij}$, thus selecting the set of matches that minimizes the total cost. S^* univocally represents a set of tracks: a track with m detections $T_k = \{r_{i_1}, r_{i_2}, \ldots, r_{i_m}\}$ is defined by the sequence of matches of Eq. 1.

$$s_{i_1 i_2} = 1,\ s_{i_2 i_3} = 1,\ \ldots,\ s_{i_{m-1} i_m} = 1 \tag{1}$$

It's important to augment C to model track initiation and ter-

This work was supported by BRISA, Auto-estradas de Portugal, S.A.

mination, otherwise solutions with these events will be penalized heavily. A simple solution is presented in Eq. 2. The sub-matrix C_{init} is presented in Eq. 3, and C_{term} follows the same structure. $c_{init,k}$ and $c_{term,k}$ hold the costs of initializing and terminating a track at detection r_k, respectively. Matching a detection to one of these events is just as valid as matching it to another detection.

$$C' = \begin{bmatrix} C & C_{term} \\ C_{init} & \mathbf{0}_{n\times n} \end{bmatrix} \tag{2}$$

$$C_{init} = \begin{bmatrix} c_{init,1} & \infty & \cdots & \infty \\ \infty & c_{init,2} & \cdots & \infty \\ \vdots & \vdots & \ddots & \vdots \\ \infty & \infty & \cdots & c_{init,n} \end{bmatrix} \tag{3}$$

This formulation is superior to the one used by Stauffer et al. [5], which only allows initialization or termination of a single track per source or sink (discrete locations of entry and exit), in the whole sequence, due to the no-overlap restriction. The formulation above allows initialization and termination of an arbitrary number of tracks.

2.2. Probabilistic framework

While the approach described previously is effective, the costs are a bit awkward to work with, since there is no clear way of combining different metrics (such as color and position differences) in a single cost. Huang et al. [6] suggested a probabilistic formulation that maximizes a sum of log-likelihoods, and by learning the distribution of each metric with training data, the fusion problem is solved in a natural way. We use their joint probability P_{ij}, which is simply a product of likelihoods, assuming mutual independence. The metrics used in our tracker are position and size differences (modeled using 2D Gaussians), time difference (modeled with a Bernoulli distribution), and Region Covariance Matrix (RCM) dissimilarity (modeled with the half-normal distribution). RCMs [8] were chosen because they can fuse naturally a number of image features; in our case, color, gradient and spatial coordinates information.

The model can also use *a priori* information of the confidence in each detection, the likelihood P_i^+ (while the likelihood of a false alarm is $P_i^- = 1 - P_i^+$). Based on the formulation of Huang et al., we model the likelihood of a track S_k with the Markov chain of probabilities shown in Eq. 4. A track has initiation and termination terms, detection likelihoods, and link terms between them.

$$\begin{aligned} P(S_k) &= P_{init,i_1} P_{i_1}^+ P_{i_1 i_2} P_{i_2}^+ P_{i_2 i_3} (\ldots) \\ &\quad P_{i_{m-1}}^+ P_{i_{m-1} i_m} P_{i_m}^+ P_{term,i_m} \end{aligned} \tag{4}$$

The objective function is to maximize $\prod P(S_k) \prod P_i^-$, where the first product is over all tracks S_k, and the second product is over all rejected detections r_i, which don't belong to any track.

This can be transformed into the costs presented in Eq. 5 to 7. As shown in [6], solving the matching problem with these costs is equivalent to optimizing the probabilistic objective function.

$$c_{ij} = \begin{cases} -\log P_i^-, & \text{if } i = j \\ -\log \left[\sqrt{P_i^+} P_{ij} \sqrt{P_j^+} \right], & \text{otherwise} \end{cases} \tag{5}$$

$$c_{init,k} = -\log \left[P_{init,k} \sqrt{P_k^+} \right] \tag{6}$$

$$c_{term,k} = -\log \left[P_{term,k} \sqrt{P_k^+} \right] \tag{7}$$

2.3. Why a sliding window can't be used directly

The main issue with methods like the above is that they can only be applied to contained video segments, since they require full information to operate. This rules out important applications such as continuous video surveillance. A sliding window is a popular approach for continuous operation [9, 2] that restricts processing to a rolling buffer of the most recent data; for instance, within the last w frames.

Consider what would happen if the proposed tracker was applied to the detections of frames 1 to w, obtaining tracks $\mathcal{T}_1$; then to frames 2 to $w+1$ to obtain tracks $\mathcal{T}_2$, and so on, in a manner similar to most sliding window approaches. Intuitively, the tracks in $\mathcal{T}_1$ and $\mathcal{T}_2$ should mostly overlap. By visual inspection, a human can easily match these tracks, and heuristic methods could do the same job. However, the assumption that tracks in consecutive windows mostly overlap is wrong, since the optimal solution for one window *can* be very different from the other (consider switched identities, new objects, deciding whether each detection is a false alarm or not). More formally, global solutions of similar problems aren't necessarily similar.

Worse, even though we just matched detections in each window, we still have to match them *among* windows, which is another matching problem. Instead of clumsily gluing global matching to the sliding window approach, we integrate them seamlessly through the Dynamic Hungarian Algorithm.

3. TRACKING IN STREAMED VIDEO

3.1. The Dynamic Hungarian Algorithm

Given the facts discussed in the previous section, maintaining the optimal set of tracks as the sliding window advances seems like a daunting task. However, Mills-Tettey et al. [10] have shown recently that the same basic building blocks of the Hungarian algorithm can be used to *repair* a solution of a matching problem in the presence of changed costs. Although their work was intended mainly for logistics problems, we found that it serves as a sound theoretical and practical framework for our purposes. This Dynamic Hungarian Algorithm can repair p rows or columns of costs in $O(pn^2)$ time, where n is the number of detections, typically orders of magnitude greater than p. Re-building the same solution from scratch would take $O(n^3)$ time. We will now show how it applies to our work.

The speed gain is not the focus of this extension. Tracking with global methods in streamed video is simply **not** possible without it.

3.2. Overview

We will refer to each state of the sliding window as an *iteration*. The detection r_i is said to be inside the window at iteration k, defined by the time interval $[w_{start,k},\ w_{end,k}]$, if its time instant t_i is inside that interval. The state of the tracker is simply the set of detections in the window, $\mathcal{R}_k$, and the corresponding cost matrix C_k. When transitioning to the iteration $k+1$, due to the movement of the window, detections $\mathcal{R}_{new,k+1}$ that are ahead in time will enter the window, and detections $\mathcal{R}_{rem.,k+1}$ will leave it. We will repair and maintain the optimal solution M_k^* of each iteration as this happens.[1] Section 3.5 describes the storage process and discusses optimality.

[1] The window can advance any number of frames with each iteration, as long as there is overlap with the previous window. This doesn't affect the solution.

3.3. Integration of new detections

Let $n = |\mathcal{R}_{k-1}|$ and $m = |\mathcal{R}_{new,k}|$. First, we update the buffer with the new detections, as in Eq. 8. The cost matrix is extended with trivial infinite costs in Eq. 9, which mean that no matches are possible for the new detections. Since the previous solution M^*_{k-1} doesn't match the new detections, it is still optimal for this cost matrix.

$$\mathcal{R}'_k = \mathcal{R}_{k-1} \cup \mathcal{R}_{new,k} \tag{8}$$

$$C^1_k = \begin{bmatrix} C_{k-1} & \infty_{n\times m} \\ \infty_{m\times n} & \infty_{m\times m} \end{bmatrix} \tag{9}$$

Of course, the costs related to the new detections are not correct. The correct costs are given by Eq. 10, where $C(\mathcal{I};\, \mathcal{J})$ are the costs of matching detections in the set $\mathcal{I}$ to detections in the set $\mathcal{J}$ (ie, $C(\mathcal{I};\, \mathcal{J}) = [c_{ij}]$, $r_i \in \mathcal{I}$, $r_j \in \mathcal{J}$). The elements in the lower-left block are still infinite since they are the costs of matching new detections to previous detections, which cannot happen because they always occur later in time.

$$C^2_k = \begin{bmatrix} C_{k-1} & C(\mathcal{R}_{k-1};\, \mathcal{R}_{new,k}) \\ \infty_{m\times n} & C(\mathcal{R}_{new,k};\, \mathcal{R}_{new,k}) \end{bmatrix} \tag{10}$$

The Dynamic Hungarian Algorithm can be used to update the solution by changing the costs from C^1_k to C^2_k. Only the right-most m columns must be updated.

3.4. Removal of detections

Just as new detections enter the window, others will be left behind. We denote these $\mathcal{R}_{rem.,k}$, and $p = |\mathcal{R}_{rem.,k}|$. To maintain the optimal solution, the approach is very similar to the one in the previous section. The Dynamic Hungarian Algorithm updates the solution based on the changed costs, from C^2_k to C^3_k, shown in Eq. 11, where $b = m + n - p$.

$$C^3_k = \begin{bmatrix} \infty_{p\times p} & \infty_{p\times b} \\ \infty_{b\times p} & C_k \end{bmatrix} \tag{11}$$

Only the upper p rows have to be updated. Notice that this way we finally obtain C_k (it's embedded in C^3_k), which will be re-used in iteration $k+1$. This can also be interpreted as removing the rows and columns from C^2_k that correspond to the removed detections. The detections are removed from the buffer in Eq. 12, and $\mathcal{R}_k$ is now ready to be re-used over the next iteration.

$$\mathcal{R}_k = \mathcal{R}'_k \setminus \mathcal{R}_{rem.,k} \tag{12}$$

This concludes the process of updating the solution dynamically as the window moves forward. This formulation ensures that the matching is optimal at the end of every iteration.

3.5. Track commitment and optimality

The matches in the buffer represent only the *current best estimate* of the tracks, and may change in future iterations. This means that no commitment can be made (ie., no match can be considered permanent). But obviously some matches have to be committed at some point; otherwise the options under consideration (the size of the window) will grow indefinitely and we'd be no better than with the previous global method.

The only matches that are no longer subject to change are the ones left behind by the sliding window, $S^*_{rem.,k}$ (corresponding to $\mathcal{R}_{rem.,k}$). They are the ones most distant (in time) from the new detections, and with a pseudo-Markovian assumption that detections farther in time are more likely to be independent[2], the oldest matches are also the ones less likely to need revision.

The removed matches $S^*_{rem.,k}$ are stored continually as the final solution. This also means that larger time windows allow for greater confidence in the stored matches, since they are even less likely to need revision when new data arrives, using the same assumption. It entails a certain degree of trade-off between number of computations and quality of output, controlled by the size of the time window.

Probably the event that is most likely to violate the Markovian assumption is the occlusion of an object for an extended period. If an occlusion period is greater than the window size, the tracker cannot match the last known detection of the object to the new detection when it reappears. This should be easy to understand, as the tracker "forgot" the last state of the object. To account for long occlusions the window size should be large.

Fortunately, the small amount of computations involved with each update allows for very large window sizes to be used (when compared to methods such as MHT), allowing the tracker to compensate for very long occlusions (up to several seconds or minutes). Also, given the dynamic update of the solution, the solution is *always* optimal for the time window over *all* iterations.

3.6. Optimal tracks

The stored matches are given in Eq. 13. They can be stored to a growing buffer, for example in a mass storage device. The detections they refer to, $\mathcal{R}_{rem.,k}$, can be stored in the same way. Tracks can be recovered as sequences of matches, as explained in Section 2.1.

$$S^* = S^*_{rem.,1} \cup S^*_{rem.,2} \cup \ldots \cup S^*_{rem.,N} \tag{13}$$

Table 1: Results for each video sequence.

Video Sequence	Tracked	Hit Rate	Pos. Error
Corridor	7 / 7	0.9825	0.2878
EnterExit...1cor	5 / 5	0.9688	0.1988
WalkBy...1front	5 / 5	0.9929	0.1726
EnterExit...1front	4 / 4	0.9797	0.1199
Highway	47 / 54	0.9676	0.1893
WalkBy...1cor	18 / 20	0.8781	0.2147

4. RESULTS

We applied the tracker to a number of datasets, obtaining the results presented in Table 1. We pair each detection with the closest detection in the ground truth, and consider it correct if their overlap area is over 50%. A track is considered correct if more than 90% of its detections are also correct. The second column shows the number of correct tracks, as well as the total number of tracks in the ground truth. The third column presents the average of the ratio of correct detections for the correct tracks. The normalized position error appears in the fourth column. It's simply the average of the euclidean distances between detections and their matches in the ground truth. The distances are normalized to the lengths of the diagonals of the ground truth bounding boxes, making this measure invariant to size.

[2]The well-known Markovian assumption is that a state is independent of all states except for the one *immediately* preceding it. Our assumption is much more loose. We only require it to be independent of states preceding it by a sufficiently large amount of time (or number of state transitions).

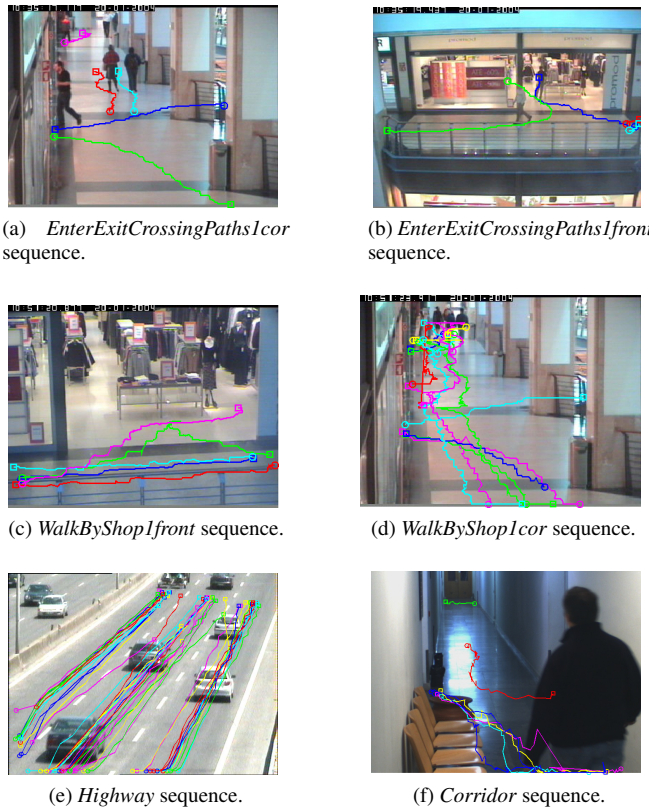
The resulting tracks can be seen in Figure 1. The trajectory of each object on the ground appears in a different color (colors are cycled when the number of objects is large). The tested videos are from the CAVIAR dataset[3], using the supplied labellings as detections, except for the *Corridor* and *Highway* sequences. The *Corridor* video was captured especially for our own use, and the *Highway* video was processed with a detector based on foreground segmentation, currently under development at our laboratory.

The video *WalkByShop1cor* (CAVIAR) is the most challenging, since it is rather long (2360 frames) and has many detections (11318). The cost matrix for such a sequence, using current global matching schemes, would have more than 500 million elements. Long occlusions, like ones of the blue and cyan tracks, respectively in Figures 1a and 1d, passing behind a pillar, are handled correctly.

The tracker's parameters are left unchanged for all test sequences, including the *Highway* video, which is a testament to the robustness of this method. The estimated probability of entry or exit in the scene (P_{init} and P_{term}), in the form of a spatial map, was provided to the tracker as *a priori* knowledge, but this map could easily be learned over time as shown in [6]. The covariances of the likelihood models referred in Section 2.2 were obtained in a training stage, using ground truth data from other videos in the same datasets. A time window of just 60 frames sufficed for all tests, but consistent results were obtained with much larger time windows, to deal with more severe occlusions, still in real-time.

The probabilistic nature of the tracker allows it to reject many mistakes of the detector. The tracker performs very well with the good quality detections of the CAVIAR videos, but it still performs reliably with the lower quality detections of the other datasets (*Corridor* and *Highway*). This is a very important characteristic, since reliable detection is still out of the reach of the current state-of-the-art detectors, without any specific parameter tuning.

The main contribution of this paper, continuous operation on streamed input, is demonstrated by applying it to very long sequences with no impact on performance.



(a) *EnterExitCrossingPaths1cor* sequence.

(b) *EnterExitCrossingPaths1front* sequence.

(c) *WalkByShop1front* sequence.

(d) *WalkByShop1cor* sequence.

(e) *Highway* sequence.

(f) *Corridor* sequence.

Fig. 1: Resulting tracks for each video, superimposed on an example frame. Ground positions are estimated as the bottom-center of each detection's bounding box. Humans and vehicles were tracked under different conditions, using the same set of tracker parameters.

5. CONCLUSION

We set out to solve one of the glaring omissions of global matching trackers, which is their inability to operate continuously with a streamed video input. Without this addition, optimal matching methods can only be applied *a posteriori* to track objects in a contained video, with complete knowledge of the scene and very high computational costs. Nonetheless, such algorithms have a tremendous advantage over most other methods, which are limited to local decisions instead of globally optimal ones. Based on a sound theoretical framework that guarantees optimality of the matching at all times, we proposed an update procedure that can be applied to a sliding window of detections over time. This finally enables this class of algorithms to work over *continuous* video streams, a critical requirement of video surveillance applications. In addition, encouraging results demonstrate the effectiveness of this technique, its flexibility under various difficult conditions, and even its robustness to its own parameters.

6. REFERENCES

[1] K. Okuma, A. Taleghani, N. De Freitas, J. J Little, and D. G Lowe, "A boosted particle filter: Multitarget detection and tracking," *Lecture Notes in Computer Science*, 2004.

[2] M. Betke, D. E. Hirsh, A. Bagchi, N. I. Hristov, N. C. Makris, and T. H. Kunz, "Tracking large variable numbers of objects in clutter," *Proceedings of the IEEE Computer Society*, 2007.

[3] K. Shafique and M. Shah, "A noniterative greedy algorithm for multiframe point correspondence," *IEEE transactions on pattern analysis and machine intelligence*, 2005.

[4] D. B Reid, "An algorithm for tracking multiple targets," *IEEE Transactions on Automatic Control*, 1979.

[5] C. Stauffer, "Estimating tracking sources and sinks," in *Computer Vision and Pattern Recognition Workshop*, 2003.

[6] C. Huang, B. Wu, and R. Nevatia, "Robust object tracking by hierarchical association of detection responses," *Proceedings of the 10th European Conference on Computer Vision*, 2008.

[7] Ding-Zhu Du and Panos M. Pardalos, *Handbook of Combinatorial Optimization - Supplement Volume A*, Springer, 1999.

[8] F. Porikli, O. Tuzel, and P. Meer, "Covariance tracking using model update based on lie algebra," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[9] Q. Yu and G. Medioni, "Multiple-target tracking by spatiotemporal monte carlo markov chain data association," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.

[10] G. A Mills-Tettey, A. Stentz, and M. B Dias, "The dynamic hungarian algorithm for the assignment problem with changing costs," 2007.

[3] http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/