DYNAMIC GLOBAL OPTIMIZATION FRAMEWORK FOR REAL-TIME TRACKING

João F. Henriques, Rui Caseiro and Jorge Batista
Institute of Systems and Robotics, Department of Electrical and Computer Engineering
University of Coimbra, Portugal

Keywords: Visual surveillance, Tracking, real-time, Dynamic Hungarian algorithm, Region covariance matrices.

Abstract: Tracking is a crucial task in the context of visual surveillance. There are roughly three classes of trackers: the classical greedy algorithms (based on sequential modeling of targets, such as particle filters), Multiple Hypothesis Tracking (MHT) and its variants, and global optimizers (based on optimal matching algorithms from linear programming). We point out the shortcomings of all approaches, and set out to solve the only gaping deficiency of global optimization trackers, which is their inability to work with streamed video, in continual operation. We present an extension to the new Dynamic Hungarian Algorithm that achieves this effect, and show tracking results in such different conditions as the tracking of humans and vehicles, in different scenes, using the same set of parameters for our tracker.

1 INTRODUCTION

The past few years have seen an increased interest in the development of automatic surveillance systems. Many approaches are based exclusively on the interpretation of color camera images (as opposed to, for example, multi-sensor networks), since it would allow relatively easy integration with existing CCTV camera networks. Security and monitoring applications require that a system is capable of (a) detecting people, (b) tracking them while maintaining their true identities over difficult situations such as occlusion and appearance changes, and (c) identifying and reacting to their behavior. The focus of this work is the second task.

Many approaches have been proposed since automatic tracking became feasible, but only recently has a significant breakthrough been made: the use of global optimization methods.

Trackers of this type have an enormous advantage over classical trackers, since they rely on the use of global information. Trackers based on Kalman filters, particle filters (Palaio and Batista, 2008; Okuma et al., 2004) and other ad-hoc greedy algorithms (Betke et al., 2007; Shafique and Shah, 2005) are all examples of classical trackers. Classical tracking systems are *greedy*, in that they're limited to information about the current frame, and a summary of the information from previous frames (i.e., the filters' states, a list of tracked objects, etc). This often leads to trapping in local minima of the functions they try to optimize, and drifting when faced with ambiguities that can't be resolved immediately. The popular Multiple Hypothesis Tracker (MHT) (Reid, 1979) improves on this, and can be seen as a transitional step between greedy methods and global optimization. However, the combinatorial explosion from all the different possibilities under consideration limits its window of operation to no more than a few frames. The global methods we refer to, based on the Hungarian algorithm (Stauffer, 2003; Taj et al., 2007), enjoy a much smaller computational complexity, since they take advantage of the sub-structure of the matching problem; their worst-case running time is $O(n^3)$, where n is the number of detections over all frames under consideration (instead of exponential). The earliest uses of the Hungarian algorithm in tracking applications were the matching of objects from different cameras with disjoint views (Huang and Russell, 1997), where it remains a popular algorithm (Javed et al., 2003), but since then it has been generalized for tracking within a single camera.

One way to reduce n substantially, improving running times by an order of magnitude, is precomputing *tracklets* using a conservative strategy (described in Section 2.1). Evidence of this sort of reasoning can be found in several other works, under different names. Kanade et al. (Li et al., 2008) uses a *track compiler* to produce *track segments*, associated later by a *track linker* (these terms correspond, respec-

Henriques J., Caseiro R. and Batista J. (2010).
DYNAMIC GLOBAL OPTIMIZATION FRAMEWORK FOR REAL-TIME TRACKING.
In *Proceedings of the International Conference on Computer Vision Theory and Applications*, pages 207-215
DOI: 10.5220/0002823502070215

tively, to our conservative association, tracklets and optimal association). Stauffer (Stauffer, 2003) refers to the later as *track stitching*. Both Stauffer and Nevatia (Huang et al., 2008) refer to the track segments as *tracklets*.

An inherent issue of global optimization is easy to understand: in a realistic scenario, we obviously don't have access to the whole video to analize it globally; instead, a continuous video stream is received over time, and we wish to obtain tracking results as immediately as possible. To this date, this promising class of methods hasn't been able to make the leap from global analysis of a single video segment to analysis of continuous video streams. Our work intends to bridge this gap, through the method outlined in Section 4.

This paper is organized as follows. Section 2 describes the tracking framework based on a probabilistic formulation of the problem, which is then solved by the Hungarian Algorithm. Section 3 showcases the appearance descriptor of our choice, the Region Covariance Matrix (RCM). Section 4 presents the extension of the Dynamic Hungarian Algorithm to deal with a sliding window, enabling its use in continuous, streamed video, as opposed to small video segments as has been the case in previous work. Finally, Sections 5 and 6 show, respectively, the experimental results and our conclusions.

2 TRACKING METHODOLOGY

Our tracking method starts with a stripped-down implementation of the hierarchical tracker proposed by Nevatia et al. (Huang et al., 2008). Their work follows the recent trend of computing association scores between all pairs of detections and using the Hungarian algorithm to create a *matching* between them [1], thus obtaining a set of tracks, in a way that optimizes the association scores. They computed the associations progressively, through a hierarchy of low, middle and high-level association schemes; the basic framework for our tracker is adapted from theirs, and will be described in this section.

2.1 Conservative Association

Recall that the main objective is to associate (match) each detection to another one, optimizing some association criteria. In a typical scene, there's a good number of associations that are straightforward to compute. For example, a person walking down a corridor alone without any occlusion will yield a set of detections with high association scores, and no other detections should have equally high scores towards those detections. In such cases matching is easily computed and is unambiguous, by a process we call *conservative association*. This turns out to be an efficient optimization, relieving the Hungarian algorithm of this duty (the algorithm's running time is $O(n^3)$, with n the number of detections).

2.1.1 Conservative Strategy

We denote r_i as a detection response, which may contain characteristics such as position, frame index, and appearance properties. Instead of arbitrary scores, it makes sense to maximize association probabilities, so these will be used throughout the text. The aim of this first take on matching is to consider matches that have a high association probability (higher than an arbitrary threshold θ_1), but only if there is no other conflicting match; that is, all other matches involving these two detections have lower probabilities (by at least θ_2). This is defined in (1), where $P_{link}(r_i|r_j)$ is the association probability between detections r_i and r_j.

$$\begin{cases} P_{link}(r_i|r_j) > \theta_1 \\ \min\Big(P_{link}(r_i|r_j) - P_{link}(r_k|r_j), \\ \qquad P_{link}(r_i|r_j) - P_{link}(r_i|r_k)\Big) > \theta_2, \\ \qquad \forall r_k \in R - \{r_i, r_j\} \end{cases} \tag{1}$$

2.1.2 Association Probabilities

The association probabilities can be computed through (2), which is simply the joint probability of three probabilities of identity, called *affinities*. $A_\delta(r_i|r_j), \delta \in \{p,s,a\}$ are position, size and appearance affinities (described in the next paragraph), and t_k is the frame index of the occurrence of detection r_k. Note that the only way for an association probability to be non-zero is for the second detection to appear exactly one frame after the first. This is part of the conservative strategy, as occlusions (i.e., frame gaps between detections) are not resolved at this stage.

$$P_{link}(r_i|r_j) = \begin{cases} A_p(r_i|r_j)A_s(r_i|r_j)A_a(r_i|r_j), & \text{if } t_j - t_i = 1 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The position difference between two detections is modeled through a two-dimensional Gaussian distribution so the position affinity can be obtained from the positions of two detections, p_i and p_j, as $G(p_i -$

[1] In the tracking context, a matching indicates, for each detection, which one comes next.

$p_j; 0, \Sigma)$ (a Gaussian with zero mean and covariance matrix obtained from sample data). Likewise for the size affinity and appearance affinity. The later is obtained from the dissimilarity metric described in Section 3, this time using a single-dimensional Gaussian distribution.

The result of this stage is a set of early matches, that by no means have to include all of the detections. They represent a disjointed set of track segments, called tracklets. These tracklets can be further associated by the Hungarian algorithm as described in Section 2.2.

2.2 Optimal Association

As was stated before, the Hungarian algorithm (Kuhn, 1955) computes an optimal matching of detections. Specifically, we can assign each possible match (T_i, T_j) a cost c_{ij}, through a cost matrix C; the algorithm will compute the set of independent matches that minimizes the sum of all costs.

2.2.1 MAP Formulation

In (Huang et al., 2008), the objectives of tracking are stated as the MAP problem (3), where $\mathcal{S}$ is a set of tracks, $\mathcal{S}^*$ is the optimal set of tracks, and $\mathcal{T}$ is the set of all *tracklets*, through direct application of Bayes' theorem.

$$\begin{aligned} \mathcal{S}^* &= \arg\max_{\mathcal{S}} P(\mathcal{S}|\mathcal{T}) = \arg\max_{\mathcal{S}} P(\mathcal{T}|\mathcal{S})P(\mathcal{S}) \\ &= \arg\max_{\mathcal{S}} \prod_{T_i \in \mathcal{T}} P(T_i|\mathcal{S}) \prod_{S_k \in \mathcal{S}} P(S_k) \end{aligned} \quad (3)$$

The conditional probability of a tracklet given the set $\mathcal{S}$ depends on its inclusion in the solution, and is modeled by a Bernoulli distribution from the hit rate β of the detector and the number of elements $|T_i|$ in the tracklet (4).

$$P(T_i|\mathcal{S}) = \begin{cases} P_+(T_i) = \beta^{|T_i|}, & \text{if } \exists S_k \in \mathcal{S}, T_i \in S_k \\ P_-(T_i) = (1-\beta)^{|T_i|}, & \text{otherwise} \end{cases} \quad (4)$$

Finally, the prior probability of an association of tracklets S_k is a Markov Chain with initialization and termination probabilities P_{init} and P_{term}, and a series of link probabilities covering all the tracklets in the sequence (5).

$$\begin{aligned} P(S_k) &= P_{init}(T_{i_0}) P_{link}(T_{i_0}|T_{i_1}) (\ldots) \\ & \quad P_{link}(T_{i_{(n_k-1)}}|T_{i_{(n_k)}}) P_{term}(T_{i_{(n_k)}}) \end{aligned} \quad (5)$$

The link probabilities, similarly to Section 2.1.2, are given by the joint probabilities of motion (A_m), temporal (A_t) and appearance (A_a) affinities, as shown in (6). Due to space limitations we won't get into much details about the motion and temporal components, as this is explained thoroughly in (Huang et al., 2008). The temporal component is modeled through a Bernoulli distribution according to the time gap between two tracklets. The motion affinity is obtained by projection through the time gap, assuming a constant velocity model obtained with a Kalman filter, and finally the projected positions are modeled with Gaussians (similarly to the position affinity described in Section 2.1.2). The appearance component is calculated in the same way as in Section 2.1.2.

$$P_{link}(T_i|T_j) = A_m(T_i|T_j) A_t(T_i|T_j) A_a(T_i|T_j) \quad (6)$$

2.2.2 Cost Matrix Definition

The above formulation can be decomposed into the elements of a cost matrix (7), in such a way that the optimal matching corresponds to the solution to the MAP problem.

$$C = \left[\begin{array}{cccc|cccc} c_{11} & c_{12} & \cdots & c_{1n} & f_1 & \infty & \cdots & \infty \\ c_{21} & c_{22} & \cdots & c_{2n} & \infty & f_2 & \cdots & \infty \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{c2} & \cdots & c_{nn} & \infty & \infty & \cdots & f_n \\ \hline i_1 & \infty & \cdots & \infty & 0 & 0 & \cdots & 0 \\ \infty & i_2 & \cdots & \infty & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \infty & \infty & \cdots & i_n & 0 & 0 & \cdots & 0 \end{array}\right] \quad (7)$$

The off-diagonal elements of the upper-left block represent regular tracklet-to-tracklet matches. A sequence of matches of this nature will constitute a track. A match to a diagonal element represents a false alarm: tracklets in this situation are left out of the final set of tracks and ignored entirely. Matches to the diagonal elements of the upper-right block terminate tracks, while matches to the diagonal elements of the bottom-left block initiate tracks. The bottom-right block is unused and any match occurring here incurs no penalty.

$$c_{ij} = \begin{cases} -\log P_-(T_i), & \text{if } i = j \\ -\log\left[\sqrt{P_+(T_i)} P_{link}(T_i, T_j) \sqrt{P_+(T_j)}\right], & \text{otherwise} \end{cases}$$

$$\begin{aligned} i_k &= -\log\left[P_{init}(T_k)\sqrt{P_+(T_k)}\right] \\ f_k &= -\log\left[P_{term}(T_k)\sqrt{P_-(T_k)}\right] \end{aligned}$$

2.2.3 Optimal Matching and Obtaining the Objects' Tracks

The Hungarian algorithm (Kuhn, 1955) is well-known and described extensively in the literature (Ahuja, 2008). It finds the optimal matching M^* given the cost matrix C, in the form shown in (8).

$$M^* = \{(T_i, T_j) \,|\, i, j \in 1, \ldots, n\} \tag{8}$$

For this result to be meaningful in the tracking context, we need to obtain a set of tracks, each one composed of a sequence of tracklets. This can be done by resorting to a connected components algorithm.

Given a $2n \times 2n$ matrix C, we're only interested in the first n elements of the matching M^*, which represent matches between tracklets, and false alarms (in the form (T_i, T_i)). An $n \times n$ adjacency matrix A^* can be constructed as described in (9).

$$A^* = \{a_{ij}\},\ a_{ij} = \begin{cases} 1, & \text{if } (T_i, T_j) \in M^* \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

Finding the connected components in the graph represented by A^*, one gets a set of independent tracks $\mathcal{S}^*$ as required. The tracks that contain only one element are the false alarms (the (T_i, T_i) matches) and can be rejected at this point.

3 REGION COVARIANCE MATRICES

The most commonly accepted object descriptor for video surveillance applications is the color histogram (Okuma et al., 2004; Javed et al., 2003), as it is discriminative in many situations and is relatively robust against object pose changes. However, it doesn't take the object's geometry into account, nor the spatial distribution of the colors it attempts to model. These features would be desirable as they would allow us to distinguish objects with similar colors but differing spatial distributions, for instance. The inclusion of more features into the histogram rapidly increases its storage and computation overhead, and increases the difficulty of working with the data due to the "curse of dimensionality". A descriptor that addresses these concerns is the Region Covariance Matrix (RCM). It has been used as a local descriptor for cascade-based detectors (Tuzel et al., 2008) and as a more generic object descriptor for tracking (Porikli et al., 2006). It has been reported to be able to match objects with moderate variations in pose and geometry in this last study.

An RCM compactly aggregates color, gradient and spatial information about a region. Consider a function $\Phi(I, x, y)$ that obtains these features for each pixel of an image I. Use it to create a $W \times H \times d$ tensor of all features, F. The d-dimensional points inside a given region $R \subset F$ are $\{z_i\}_{i=1\ldots S}$. Then, the corresponding RCM is the $d \times d$ matrix given by (10), where μ_R is the mean of those points.

$$C_R = \frac{1}{S-1} \sum_{i=1}^{S} (z_i - \mu_R)(z_i - \mu_R)^T \tag{10}$$

An RCM has a number of advantages when compared to many other descriptors. An RCM encodes the variance of every feature and correlations between all pairs of features. It naturally acts as an averaging filter over all samples, eliminating some forms of noise; it rejects the mean of the encoded features, which means that it's naturally invariant to illumination variations, in the case of the color channels, and has a similar invariance towards the other features; and since different regions always yield RCMs of the same size, it can be used to compare regions of different sizes. Probably the best advantage of RCMs is their ability to fuse radically different features without resorting to artificial weighting of their contributions.

3.1 Features Set

The features that are aggregated into an RCM for the task of object detection, as suggested in (Tuzel et al., 2008), represent the position of the samples (x, y), and the first (I_x, I_y) and second-order spatial derivatives (I_{xx}, I_{yy}) of the image intensities, as shown in (11).

$$f_1 = \begin{bmatrix} x & y & |I_x| & |I_y| & \|I'\| & |I_{xx}| & |I_{yy}| & \angle I' \end{bmatrix}^T$$

$$\|I'\| = \sqrt{I_x^2 + I_y^2},\ \angle I' = \arctan \frac{|I_x|}{|I_y|} \tag{11}$$

In the case of object tracking, since the use of color is well suited for discrimination between objects, we reduce the number of spatial derivatives and add color information. We also replaced the (x, y) positions of the samples by four spatial functions, $\rho_{i=1\ldots4}$. The resulting vector is (12), where $|I_{xy}|$ is the Laplacian operator (second order spatial derivative) and R, G, B are the color channels.

$$f_2 = \begin{bmatrix} \rho_1 & \cdots & \rho_4 & |I_x| & |I_y| & |I_{xy}| & R & G & B \end{bmatrix} \tag{12}$$

Since an RCM models correlations between the selected features, we hypothesized that correlating

features with functions that have high values in certain regions would be more meaningful than simply correlating them with the (x, y) positions of the samples. For tracking of walking or standing pedestrians we selected four functions that characterize three regions along the y-axis and one along the x-axis (13), where w and h are the width and height of the region R, and (x, y) is the position of the sample. The selection of the functions could be completely arbitrary, since even in the worst case, when there is absolutely no correlation between the spatial functions and the remaining features, the RCM still encodes the correlations between the non-spatial features. However, our selection was based on the simple intuition that, for the chosen class of objects, whose bounding boxes typically have a low width/height ratio, there can be noticeable discrimination between rough regions of different colors and textures along the vertical axis, but not along the horizontal axis.

$$\begin{aligned} \rho_1 &= \max(0, w/2 - |x|) \\ \rho_2 &= \max(0, h/4 - |y + h/4|) \\ \rho_3 &= \max(0, h/4 - |y|) \\ \rho_4 &= \max(0, h/4 - |y - h/4|) \end{aligned} \tag{13}$$

The use of spatial functions allows a single RCM to encode the features of more than one sub-region, inside the region of interest R. This is done instead of using multiple RCMs to characterize a single region, which would have a large impact on performance because the number of comparisons and updates would be multiplied by the number of additional RCMs.

3.2 Comparison of RCMs

Having modeled each object detection as an RCM, we need to obtain a distance metric between them in order to establish correspondences. Covariance matrices (like RCMs) belong to the space of real symmetric positive definite matrices, $Sym^+(n, \mathbb{R})$, which forms a Riemmanian manifold in the space of all matrices. Assumptions about Euclidean spaces do not hold under these conditions; for example, the space is not closed under multiplication by negative scalars, which would be necessary for the arithmetic subtraction of two covariance matrices to measure the distance between them. In (Porikli et al., 2006) a simple, closed formula that yields a measure of distance between covariance matrices is presented (14).

$$d(X, Y) = \sqrt{tr\left(\log^2\left(X^{-\frac{1}{2}} Y X^{-\frac{1}{2}}\right)\right)} \tag{14}$$

The distance formula (14) can be implemented in a way that is computationally faster by taking advantage of the fact that a matrix X in $Sym^+(n, \mathbb{R})$ can be decomposed in the form $X = UDU^T$, where U is the matrix of eigenvectors of X, and D is the corresponding diagonal matrix of eigenvalues. Then, the following identity can be used to speed up the computation of the inverse of the matrix square root of X.

$$X^{-\frac{1}{2}} = UD^{-\frac{1}{2}}U^T \tag{15}$$

Then, the matrix logarithm of $Z = X^{-\frac{1}{2}} Y X^{-\frac{1}{2}}$ can be computed fast using (16) from the decomposition of Z.

$$\log(Z) = U \log(D) U^T \tag{16}$$

Note that, when computing the distance between a fixed RCM X and a batch of other RCMs, Y_k, the value in (15) can be stored for the remainder of the operations.

3.3 Update of an RCM

In most tracking schemes, it's important to keep a good model of the appearance of each object, that best summarizes the history of the object's appearance and minimizes the impact of sudden appearance changes, which are usually erroneous. For our purposes, integrating the appearance of a new detection (an RCM) with the appearance model for that object (another RCM) is a matter of computing the *mean* of both RCMs. This can be seen as the mid-point along the geodesic in the Riemmanian manifold that connects both RCMs (here treated as *points* in the manifold). Although different, iterative methods do exist (Porikli et al., 2006), a closed formula was proposed in (Palaio and Batista, 2008) and is used here (17).

$$\bar{C} = \left(X^{\frac{1}{2}} Y X^{\frac{1}{2}}\right)^{\frac{1}{2}} \tag{17}$$

This could be applied directly in greedy tracking methods such as (Okuma et al., 2004). Since in our method tracks are the result of a closed optimization procedure, updates are not really necessary in the context of global optimization; the Hungarian algorithm only knows pairwise associations of detections or tracklets. The RCM update is used instead to summarize all the detections in a tracklet, to provide a single RCM suitable for comparison with the rest of the detections. The update scheme is described in Algorithm 1. This formulation will give a $\frac{1}{2}$ weight to the last RCM, $\frac{1}{4}$ to the second-to-last, etc, and $\frac{1}{n}$ to the first; effectively giving more importance to the most recent detections. Δt is a cut-off term: after Δt detections, the contribution of the remaining terms is considered small enough that they don't effectively matter, saving computational resources for long tracklets.

Algorithm 1: Forward appearance model for a tracklet based on successive RCM means of the tracklet's n detections.

$s := \max(n - \Delta t + 1,\ 1)$
$\bar{X} := X_s$
From $k := (s+1)$ to n
  $\bar{X} := \left(\bar{X}^{\frac{1}{2}} X_k \bar{X}^{\frac{1}{2}}\right)^{\frac{1}{2}}$
End

Algorithm 1 yields a single RCM that models the appearance of the object represented in the tracklet, at the end of the tracklet. This is useful for comparison with tracklets that occur later in time. For comparison with tracklets that occur earlier in time, a similar algorithm is used, iterating in the opposite direction, and yielding a model for the appearance of the object at the beginning of the tracklet.

4 CONTINUOUS TRACKING

4.1 Sliding Window

We propose a sliding window approach to the continuous tracking problem. This involves matching all the detections inside a time window, obtaining tracks, and moving that window forward to repeat the process as new detections arrive. There can be one such iteration per frame or every f frames. The tracks will be built on continuously, unlike other approaches that use the Hungarian algorithm and are limited to finite (and often small) video segments.

Since the window under consideration moves forward in time, there is considerable overlap among windows in consecutive iterations. Thus, the dynamic Hungarian algorithm is used, efficiently reusing part of the solution from the previous iteration.

4.2 The Dynamic Hungarian Algorithm

Mills-Tettey et al. (Mills-Tettey et al., 2007) suggested a modification to the Hungarian algorithm to update solutions in the presence of changed costs. While the Hungarian algorithm has a computational complexity of $O(n^3)$, where n is the number of vertices, updating k columns of costs using the dynamic Hungarian algorithm only has a complexity of $O(kn^2)$. We will show that moving a sliding window forward in time only requires the update of a handful of costs, making this algorithm the optimal choice for continuous tracking.

4.3 Continuous Tracking Method

Recall that r_i is a detection response, and $T_p = \{r_i | \forall i, t_i < t_{i+1}\}$ is a partial object trajectory or tracklet, composed of several detections. A degenerate tracklet may contain only one detection ($T_p = \{r_i\}$), and so the method still holds if one simply understands a "tracklet" as a "detection"[2].

4.3.1 Integration of New Data

The tracklets buffer under consideration at iteration k is denoted $\mathcal{T}_k$ ($\mathcal{T}_0 = \phi$). It is composed of all the tracklets within the sliding window, or $\mathcal{T}_k = \{T_p | \forall p, t_{end,p} \in [w_{start,k}, w_{end,k}]\}$, where the window at iteration k is defined to be between instants $w_{start,k}$ and $w_{end,k}$, and the time instant of the last detection in the tracklet T_p is $t_{end,p}$. Denote by $n_k = |\mathcal{T}_k|$ the number of tracklets in the window at iteration k. New tracklets, $\mathcal{T}_{new,k}$ ($m_k = |\mathcal{T}_{new,k}|$), are added when the window is about to advance f frames for the new iteration $k+1$. The cost matrix that holds the association costs between all pairs of tracklets in $\mathcal{T}_k$ is C_k (C_0 =empty matrix). To obtain the new cost matrix C_{k+1}, we augment the previous C_k matrix (which is $n_k \times n_k$) with the costs associated with the new tracklets, as shown in (18).[3]

The new costs are those of matching each tracklet already in the window to each new tracklet $C_{old \to new,k}$ (19), and the costs of matching new tracklets to each other $C_{new,k}$ (21). It's not possible to associate new tracklets to tracklets in the window (trajectory matches can only go forward in time), so those costs are ∞.

$$C_{k+1} = \left[\begin{array}{c|c} C_k & C_{old \to new,k} \\ \hline \infty_{m_k \times n_k} & C_{new,k} \end{array}\right] \quad (18)$$

$$C_{old \to new,k} = [c_{ij}]_{n_k \times m_k} \text{ , where} \quad (19)$$

$$i = \{i | T_i \in \mathcal{T}_k\},\ j = \{j | T_j \in \mathcal{T}_{new,k}\} \quad (20)$$

$$C_{new,k} = [c_{ij}]_{m_k \times m_k} \text{ , where} \quad (21)$$

$$i = \{i | T_i \in \mathcal{T}_{new,k}\},\ j = \{j | T_j \in \mathcal{T}_{new,k}\} \quad (22)$$

The dynamic Hungarian algorithm updates a previous matching M_k^* (with n_k matches), optimal for the previous costs C_k, to a new matching M_{k+1}^* (with

[2]This may be desirable in order to simplify the implementation, forgoing tracklets and working directly with detection responses.

[3]Here, the cost matrices are understood to not contain the initialization and termination terms (which would double their size), in order to simplify the text.

$n_k + m_k = n_{k+1}$ matches), optimal for the updated costs C_{k+1}. Since both matrices must be of the same size, we will first augment C_k with infinite cost edges in place of the new costs, as in (23).

Finally, the dynamic Hungarian algorithm will handle the transition from C'_k to C_{k+1}, updating the previous solution M^*_k to M^*_{k+1}, in the presence of m_k changed columns. These columns are the ones from n_k+1 to n_k+m_k (i.e., the right-most columns), which is apparent by comparing equations (18) and (23).

$$C'_k = \left[\begin{array}{c|c} C_k & \infty_{n_k \times m_k} \\ \hline \infty_{m_k \times n_k} & \infty_{m_k \times m_k} \end{array}\right] \tag{23}$$

Note that, although $\left|M^*_k\right| = n_k$ and $\left|M^*_{k+1}\right| = n_k + m_k$, the first n_k matches don't necessarily have to be the same. The dynamic Hungarian algorithm not only adds m_k matches to the solution, corresponding to the new tracklets, but may also change any of the existing n_k matches if required to minimize the total cost of the matching.

This process alone will yield an ever-growing set of optimal matches M^*_k as $k \to \infty$. M^*_k univocally represents a growing set of tracks for all objects on the scene, since it can be converted to a set of tracks at any point using connected components as described in Section 2.2.3.

4.3.2 Stored Matches

Given computational constraints, we know that the cost matrix can't grow indefinitely, so some matches will have to be "stored away" and never be considered again, thereby reducing the cost matrix. In practice, the stored matches will represent the full trajectory of objects observed since the system started, and can be written to any high-capacity storage media for future inspection.

In order to keep the sliding window size constant between iterations, when tracklets from f new frames are considered, tracklets from the last f frames of the window will be dropped and stored away. Let the number of tracklets from the last f frames of the window at the current iteration be p (we will drop the subscript k for clarity). We will extract a subset of p elements $M^*_{1,\ldots,p}$ from M^* and transfer it from M^* to S^* by equation (24), where S^* is the set of stored matches.

$$\begin{aligned} S^* &\leftarrow \{S^*, M^*_{1,\ldots,p}\} \\ M^* &\leftarrow M^*_s,\ \forall s \in p+1,\ldots,n+m \end{aligned} \tag{24}$$

Finally, tracklets $T_{1,\ldots,p}$ can be eliminated from $\mathcal{T}$ and from the matrix C, as shown in (25). These tracklets have been matched permanently and don't need to be considered anymore.

Algorithm 2: Continuous tracking algorithm.

$C_0 :=$ empty matrix
$\mathcal{T}_0 := \phi$
$M^*_0 := \phi$
$S^* := \phi$
$k := 0$
Do
  Advance time window f frames
  Obtain new tracklets $\mathcal{T}_{new,k}$
  $C'_k := C_k$ augmented with infinite costs, eq. (23)
  $C_{k+1} := C_k$ augmented with new costs, eq. (18)
  Dynamic Hungarian algorithm transitions $C'_k \to C_{k+1}$, updating $M^*_k \to M^*_{k+1}$
  Store matches that fall out of the window to S^*, removing them from M^*_{k+1}
  $\mathcal{T}_{k+1} := \{\mathcal{T}_k, \mathcal{T}_{new,k}\}$
  Remove tracklets that fall out of the window from $\mathcal{T}_{k+1}$
  Remove lines and columns corresponding to those tracklets from C_{k+1}
  $k := k+1$
Repeat

$$\begin{aligned} \mathcal{T} &\leftarrow \mathcal{T}_s,\ \forall s \in p+1,\ldots,n+m \\ C &\leftarrow C_{i,j},\ \forall i,j \in p+1,\ldots,n+m \end{aligned} \tag{25}$$

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
Highway	47 / 54	0.9676	0.1893
WalkBy...1cor	18 / 20	0.8781	0.2147

5 RESULTS

Quantitative results for a number of datasets are shown in Table 1. We consider a detection correct if it overlaps with the ground truth by more than 50%. The ratio of correct detections to the total number of detections in the (ground truth) track is calculated, resulting in a per-track hit rate. [4]

Then, we consider a track to be correct if its hit rate is over 90%. The second column in Table 1 shows the number of correct tracks *versus* the total from the ground truth.

The total hit rate for a video sequence is the average of the hit rates of all correct tracks, and appears in the third column.

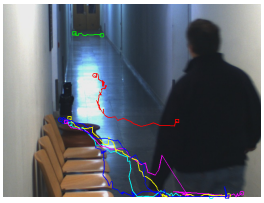
[4] Of all the ground truth tracks, the one with the highest hit rate towards a result track is assumed to be its match.

Finally, the average position error of all correct detections is presented in the fourth column. The position error of a detection is simply the euclidean distance between its position and the corresponding ground truth, divided by the length of the diagonal of the ground truth bounding box (in order to make the measure invariant to size).

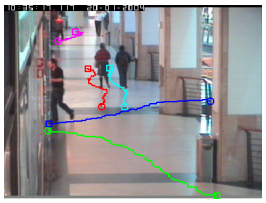
Figure 1 shows the obtained paths. We tested sequences from the CAVIAR dataset[5], and used the supplied labelings as detections. The sequence *Corridor* was captured independently for our purpose, and detections for the *Highway* sequence were obtained with an object segmentation system under development at our laboratory. Note that the sequence *WalkByShop1cor* (CAVIAR) is very challenging: there are 11318 detections over 2360 frames of video. Such a long video would require significant computational resources if analized directly with a global method; so it is the perfect test subject for our continuous tracking scheme. With a window size of 60 frames and updating the window every 20 frames ($f = 20$), we're able to track objects even in the presence of long occlusions (see the cyan track that passes behind the pillar in Figure 1e). The missed tracks are accounted for by the pack of barely visible people far away from the camera.

Note that there was **no** parameter tuning for each different sequence. The tracker is robust against its own parameters. For each scene we had to supply a map of the entry/exit locations and scene occlusions, which was done by hand but could be learned over time as in (Huang et al., 2008). We also had to find the covariance matrices for all the Gaussian models using training data, but they have similar values for all scenes; the exception is the *Highway* scene, where tracked cars have obviously different characteristics from people tracked in other videos (namely the appearance variance, which is lower).

Figure 2 presents a plot of the running time per iteration for the *WalkByShop1cor* sequence. The system is able to run in real-time, since calculations for a batch of detections are done well before the next batch arrives.

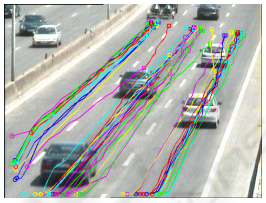


(a) *Corridor* sequence.

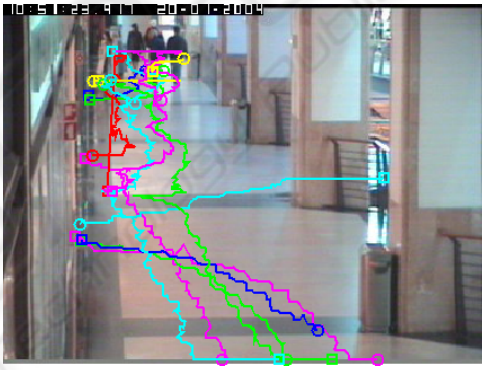


(b) *EnterExitCrossingPaths1cor* sequence.

(c) *WalkByShop1front* sequence.



(d) *Highway* sequence.



(e) *WalkByShop1cor* sequence.

Figure 1: Resulting paths of each scene, superimposed on an example frame. The positions shown are always at the bottom-center of each detection's bounding box (i.e., an estimate of its position on the ground).

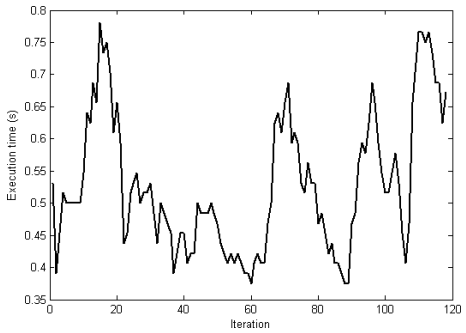


Figure 2: Execution time per iteration in the *WalkByShop1cor* sequence. Note that each iteration goes through 2 seconds of video (20 frames), but each one is processed in under 1 second in this complicated sequence.

6 CONCLUSIONS

Despite the superior performance of trackers based on global optimization methods, to this day they have been restricted to lab use due to their inherent need for complete knowledge of the scene, which is not feasible for 24 hours-a-day operation. The proposed method allows them to operate continuously. We show encouraging results from different datasets,

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

tracking both cars and pedestrians, without tuning the tracker's parameters for each set. The system is able to run in real-time, showing the flexibility of the approach and the discriminative power of Region Covariance Matrices. Hopefully we've been able to fit the missing link that will enable the adoption of global optimization methods in real-world tracking applications.

REFERENCES

Ahuja, R. K. (2008). *Network flows*. PhD thesis, Massachusetts Institute of Technology, Cambridge.

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

Huang, C., Wu, B., and Nevatia, R. (2008). Robust object tracking by hierarchical association of detection responses. In *Proceedings of the 10th European Conference on Computer Vision: Part II*, page 801.

Huang, T. and Russell, S. (1997). Object identification in a bayesian context. In *International Joint Conference on Artificial Intelligence*, volume 15, pages 1276–1283.

Javed, O., Rasheed, Z., Shafique, K., and Shah, M. (2003). Tracking across multiple cameras with disjoint views. In *Ninth IEEE International Conference on Computer Vision, 2003. Proceedings*, pages 952–957.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

Li, K., Miller, E. D., Chen, M., Kanade, T., Weiss, L. E., and Campbell, P. G. (2008). Cell population tracking and lineage construction with spatiotemporal context. *Medical Image Analysis*, 12(5):546–566.

Mills-Tettey, G. A., Stentz, A., and Dias, M. B. (2007). *The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs*. Citeseer.

Okuma, K., Taleghani, A., Freitas, N. D., Little, J. J., and Lowe, D. G. (2004). A boosted particle filter: Multitarget detection and tracking. *Lecture Notes in Computer Science*, pages 28–39.

Palaio, H. and Batista, J. (2008). A region covariance embedded in a particle filter for multi-objects tracking.

Porikli, F., Tuzel, O., and Meer, P. (2006). Covariance tracking using model update based on means on riemannian manifolds. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*.

Reid, D. B. (1979). An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854.

Shafique, K. and Shah, M. (2005). A noniterative greedy algorithm for multiframe point correspondence. *IEEE transactions on pattern analysis and machine intelligence*, 27(1):51–65.

Stauffer, C. (2003). Estimating tracking sources and sinks. In *Computer Vision and Pattern Recognition Workshop, 2003. CVPRW'03. Conference on*, volume 4.

Taj, M., Maggio, E., and Cavallaro, A. (2007). Multi-feature graph-based object tracking. *Lecture Notes in Computer Science*, 4122:190.

Tuzel, O., Porikli, F., and Meer, P. (2008). Pedestrian detection via classification on riemannian manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1713–1727.