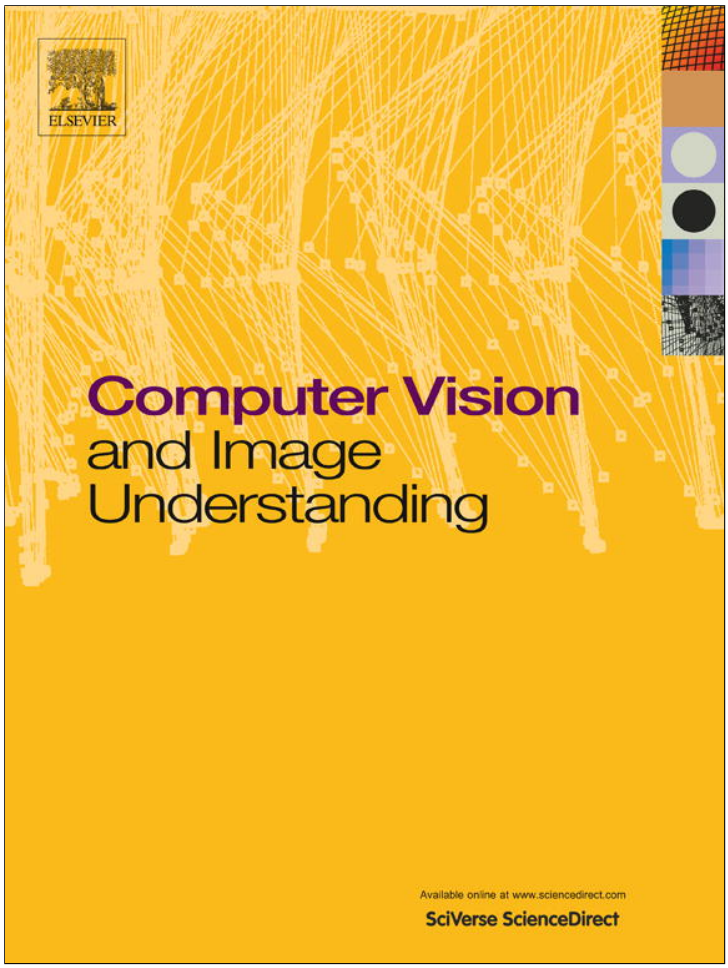

(This is a sample cover image for this issue. The actual cover is not yet available at this time.)

Computer Vision and Image Understanding 116 (2012) 967–980

Contents lists available at SciVerse ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Backtracking: Retrospective multi-target tracking ☆

W.P. Koppen *, M. Worring

Intelligent Systems Lab Amsterdam, University of Amsterdam, Science Park 904, 1098 XH Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 30 August 2010
Accepted 23 April 2012
Available online 18 May 2012

Keywords:
Surveillance
Tracking
Forensics
A*

ABSTRACT

We introduce a multi-target tracking algorithm that operates on prerecorded video as typically found in post-incident surveillance camera investigation. Apart from being robust to visual challenges such as occlusion and variation in camera view, our algorithm is also robust to temporal challenges, in particular unknown variation in frame rate. The complication with variation in frame rate is that it invalidates motion estimation. As such, tracking algorithms based on motion models will show decreased performance. On the other hand, appearance based detection in individual frames suffers from a plethora of false detections. Our tracking algorithm, albeit relying on appearance based detection, deals robustly with the caveats of both approaches. The solution rests on the fact that for prerecorded video we can make fully informed choices; not only based on preceding, but also based on following frames. We start off from an appearance based object detection algorithm able to detect in each frame all target objects. From this we build a graph structure. The detections form the graph's nodes and the vertices are formed by connecting each detection in a frame to all detections in the following frame. Thus, each path through the graph shows some particular selection of successive detections. Tracking is then reformulated as a heuristic search for optimal paths, where optimal means to find all detections belonging to a single object and excluding any other detection. We show that this approach, without an explicit motion model, is robust to both the visual and temporal challenges.

1. Introduction

Surveillance cameras are at present a widespread tool for the observation of large areas, allowing a single officer to monitor multiple locations at once. This live monitoring is mainly used for early intervention and the prevention of street crime. Another use of the video, the one this paper focuses on, is when previously recorded video is retrieved and reviewed for evidence in forensic cases or any other post-incident investigation. In such cases it is common to have large amounts of video data which need to be reviewed in their entirety.

Clearly, the most important subjects of observation are people. When we would be able to automatically find and track all recorded persons, it would greatly alleviate the exhausting process of video reviewing, but tracking all persons is a challenging task. One of the main problems of multiple person tracking is that people may *occlude* each other, in which case it is difficult for a computer to tell them apart. Another major challenge, and very common in operated surveillance video, is that the camera may *pan, tilt, and zoom*. Such operations drastically alter the perceptual location of all objects (their *xy*-position within the frame), and thus trajectories become more chaotic. This effect is amplified by the fact that, mostly to save bandwidth, many cameras use a *variable recording frame rate*. In other words, the elapsed time between any two successive frames varies. All those aspects raise severe difficulties in the prediction of object location, and thus in tracking persons.

In one of the earliest approaches towards multiple target tracking [24], alternative hypotheses about the configuration of tracks are maintained. Hence the name multiple hypothesis tracking (MHT). In MHT, a graph is constructed where each node is a hypothesis (a possible track) and the edges show how a hypothesis can change with the addition of new object detections. Thus in this graph each hypothesis that is currently not the most likely is still stored for a later moment, with the idea that future information may shed new light on which configuration is actually the most likely. Essentially this allows it to recover from tracking errors and it is particularly helpful for dealing with occlusion. However, for each new detection all leaves in the tree shaped graph need to be split into two new options: the track with the new detection added and the track without it. So with every single new detection the number of leaves doubles and the tree grows exponential in the number of detections. For large videos, such as surveillance videos, this process becomes computationally prohibitive. To alleviate the situation some works try to reduce the temporal context or prune

☆ This paper has been recommended for acceptance by R. Bergevin.

* Corresponding author.
E-mail addresses: uva@paulkoppen.com (W.P. Koppen), m.worring@uva.nl (M. Worring).

1077-3142/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.cviu.2012.04.004

the graph [22,25], but the inherent problem is storing each hypothesis separately while in fact there is just one set of underlying data; the detections. If instead we were to create a graph of the detections, then it implicitly models all hypotheses at once as it is the superset of paths through the graph. Adding a new detection is a linear operation and changing a hypothesis is just altering its path through the graph. This is the idea behind the approach taken in this paper and we will demonstrate its efficiency.

Current tracking methods that are based on sequential Monte-Carlo integration like [14,8,30] derive the direction of movement (and thus the updated object location) from a partial match of the object at its old location in the new frame. Although such methods are efficient in tracking arbitrary objects, this localization approach fails when objects move too much over successive frames. Instead of using local object-based shift, a global approach towards object tracking is achieved using optical flow fields in 17] and in [6]. Here the orientation and magnitude of movement are calculated for each pixel. Albeit being computationally costly, large object displacement is still problematic [2,6]. An interesting variation on this theme is given in [32] who compute flow over object detections rather than the pixels. Similar to linear programming [15,4] object tracks emerge from optimising (minimising) a cost function with respect to posed boundary conditions. Often however these boundaries need to be quite restrictive. For example in [4] the ground plane is manually subdivided into a grid which determines object motion (objects move only to neighboring cells) and which fixes their number by allowing entrance and exit only at designated cells. In general, approaches that focus more on the explicit detection of objects assume that object motion follows the physical laws of motion and acceleration, *i.e.* motion is not chaotic. This assumption allows the use of Kalman filters [24,23] or more general Markov models [13,28] for efficient localization of target objects. Although this is beneficial for fast object motion, it is less adequate for slow and particularly variable recording speeds as that generally causes object trajectories to behave more chaotic. So, for prerecorded video other tracking methods are needed.

In an effort to solve the problem of occlusion [3,16] incorporate the recordings of multiple cameras that film the same situation from different angles. Using a homography based approach they deal with occlusion in 3D space. Using only a single camera [7] show a combination of kernel density estimation with particle based data association to be robust against partial occlusion for articulated objects. Other single camera approaches incorporate the context around target objects: [21] learn a set of discrimination functions between foreground and background texture using linear discriminant analysis. Yang et al. [31] use data mining to detect auxiliary objects that temporarily move along with the marked target object. Although context information proves to give robustness against occlusion, both algorithms track only one single and articulated object, based primarily on appearance. An approach to explicitly model multiple object occlusion is proposed by Nillius et al. [22] and similarly by Torabi and Bilodeau [27]. Both methods differentiate between isolated tracks capturing a single target and compound tracks where multiple targets are likely to occlude each other. In doing so mutual occlusion is solved by inference on the identities joining and leaving the compound tracks. Occlusion from the scene itself however remains unsolved. The necessity of modeling compound tracks is removed when using explicit object recognition (e.g. person recognition) methods instead of blob representations. In this case an object can be 'away' for some time when it is not detected. Shafique and Shah [25], Berclaz et al. [3] and Wu et al. [29] employ a sliding time window for the explicit detection of target object reappearance after short occlusion. The window is kept quite small to minimize the rapidly growing computational overhead. Consequently it deals with occlusion of maximally this amount of time. The shared limitation of all approaches is that they cannot freely search in time for the first next clear unoccluded occurrence of each target, simply because that search space would be far too large.

In short, there are clearly two major challenges within the area of video forensics which we address in this paper; large and chaotic target object displacements and occlusion between target objects. Unlike the aforementioned methods (MHT being the exception), our method does not process video sequentially. In fact, we provide the tracking algorithm with a large temporal context, theoretically as large as the entire video. We capture detected objects in a graph structure and use A^* to find optimal paths in this graph at a minimal memory footprint. A heuristic is used that favors smooth object movement and appearance change, while allowing chaotic movement and even large gaps of missing detections because of occlusion.

In Section 2 we describe in detail our multiple person tracking algorithm. Then, in Section 3 we explore the algorithm complexity. Evaluation is performed in Section 4, followed by a discussion in Section 5. We conclude this paper with an outlook on future investigations.

2. Multiple person tracking

Our multiple person tracking algorithm builds on the availability of a person detector that can locate persons in still images – in this situation being the video frames. Our so-called *Backtracking* algorithm is composed of three stages. First it creates a *graph* connecting the person detections in each frame. Then, the most similar of those detections are used to form tracks based on the graph. These tracks may span many more frames than they hold detections, leaving gaps of skipped frames between them. Hence we term this stage of the algorithm *coarse tracking*. In the final stage the gaps are filled using a *graph search* algorithm to find the optimal path. We call this process *refinement*. Because refinement is essentially an interpolation, it is robust to the large amounts of noise present in surveillance video data, while still being applicable to the more general case of gaps in tracks. The whole process is schematically depicted in Fig. 1.

2.1. A graph of person images

We start by detecting the persons in all video frames. We use the method by Dalal and Triggs [9] as is, also trained on their person detection data set. Run frame-by-frame on the video data, the algorithm produces a set of person detections with associated likelihood ratios. The result is a set B of N detections, collected over T frames,

$$B = \{b_1, \ldots, b_N\} \tag{1}$$

where for each moment in time, t, we have the set

$$B_t \subset B, \quad t = 1, \ldots, T \tag{2}$$

The likelihood of a detection, denoted by $L(b)$, represents the system's confidence in the detection being correct with:

$$0 \leqslant L(b) \leqslant 1 \tag{3}$$

A person track then, is modeled as a concatenation of detections with the superset of B holding all conceivable tracks. To express the choice for particular tracks (and the rejection of others) we call each track a hypothesis, h. Noticing that a person has only one appearance per frame we define

$$h = [b_i, \ldots, b_j], \quad |h \cap B_t| \leqslant 1 \tag{4}$$

where $|\cdot|$ denotes the number of elements in the set.

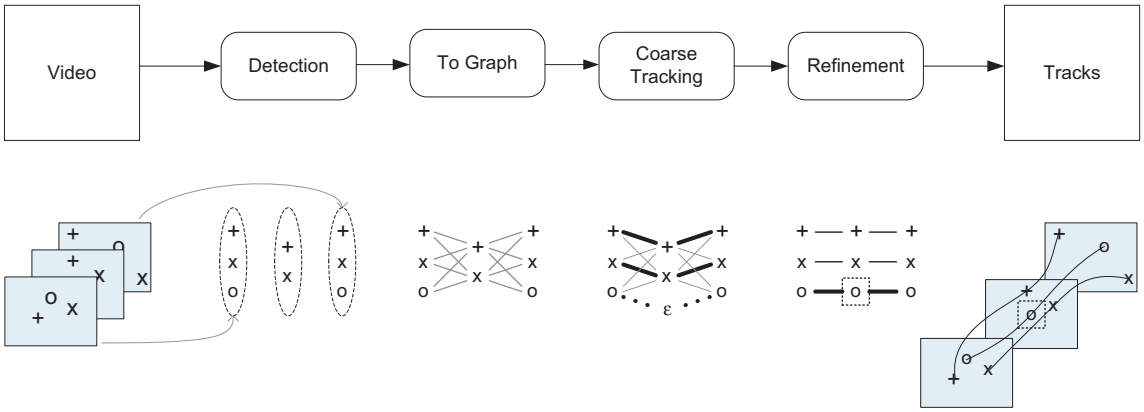

Fig. 1. Tracking process data flow. Person detection is performed for each frame separately. The detections in connected frames are linked to form a graph. A path in this graph represents a person track. The tracking algorithm proposed in this paper starts by searching for initial – coarse – paths. The empty vertex, ϵ, is used here as a placeholder so that in cases of occlusion or poor detection tracks can leap over some frames. Such gaps are then (iteratively) refined in a second stage, again using the graph.

The hypothesis space is quite large. Taking an average number of m detections per frame over n frames, there are $(m+1)^n$ different hypotheses: At every frame we can pick one of the m detections or none at all. A few boundary conditions may be excluded where a track consists of one or zero detections. These boundary cases account for $mn+1$ hypotheses and do not really reduce the hypothesis space.

To search this space efficiently we therefore represent it using a graph, where the vertices are formed of person detections and edges mark the *possibility* that two detections are of the same person. This results in a k-partite graph where vertices are adjacent if the respective detections are from successive frames.

However, this representation would be overly restrictive with respect to the set of hypotheses because it disallows *gaps*—where intermediate frames in a hypothesis are skipped, *e.g.* due to occlusion. One solution would be to extend the set of edges to connect each vertex to *all* later vertices. This is for example done in [25]. It is, however, not only unattractive for its computational complexity, it is also less meaningful to *directly* link a detection in the first frame to one in the last frame.

We propose to introduce at each frame an empty vertex, ϵ_t. Selecting this vertex means that for the moment no good detection was found just in that particular frame. In other words, it marks the case $h \cap B_t = \emptyset$. Multiple ϵ's can be visited in order to form larger gaps. This leads us to the following definition of the graph:

$$\begin{aligned} G &= (V, E) \\ V &= B \cup \epsilon \\ E &= (B_t \cup \epsilon_t) \times (B_{t+1} \cup \epsilon_{t+1}) \end{aligned} \tag{5}$$

In this representation the analogy between graph searching and object tracking is better reflected, and also the representation is very efficient because at each point in the graph we only need to consider $m+1$ options.

2.2. Coarse tracking

In order to search the graph for person tracks, we first need start and goal nodes. Operating on a subset of reliable estimates, as per $L(\cdot)$, we can use a simple matching algorithm to find those. More precisely, we find sets of two or more very similar detections with possibly large gaps in between. These sets can be regarded as outlines of tracks, or as we call them, coarse tracks, which we will now describe in detail. A pseudocode version of the algorithm is presented in Algorithm 1.

Using the detection likelihood from Eq. 3 we filter out the poor detections with a threshold, *i.e.*, only keeping detections with $L(b) > \theta$. We start with a relatively restrictive threshold so that we are confident to only include the best detections where the person is clearly in view. In track refinement, at which stage there is more information on track appearance, we will decrease the likelihood threshold to include the omitted detections.

The process of coarse tracking is an incremental approach maintaining a growing collection of tracks. We start at frame 1 and walk through the video frame by frame. At each frame we match all detections, $b \in B_t$, against the current set of tracks, H. Each track can be assigned maximally one detection and each detection can be assigned maximally once. This means that using the Hungarian algorithm [19] we find the optimal assignment, which is the assignment yielding the minimal summed cost.

We define the cost of matching a new detection to a track simply as the Bhattacharyya distance between the RGB color histograms of the new detection, b_i, and the hypothesis h_j.

$$A_{ij} = D_B(b_i, h_j) \tag{6}$$

Of course the hypothesis h_j contains more than one detection, so we need to define how the Bhattacharyya distance is computed in this case. We experimented with different weighed combinations of elements in h but found simply matching to the detection closest to the frame of reference yields the best results, suggesting that the last detection is the most representative for matching against future detections.

A natural concern with the use of RGB is that it has no invariance to lighting or other properties to enhance the matching of a person under different conditions. The general topic of color invariance spans a whole separate area of research [11], but specifically in the case of surveillance video it has been shown that invariance does actually not bring much—if any—performance gain [18]. This confirms our own experiments with the L*a*b* color space which is more similar to human perception, normalized RGB which is invariant to lightness, and HSL which models chroma and luminosity separately.

Algorithm 1. Coarse tracking

Require: $B = \{b_1, \ldots, b_N\}$

Require: $0 < \theta_0 < 1$

1: $B' = \{b \in B | L(b) > \theta_0\}$
2: $H_{open} = \{\}$
3: $H_{closed} = \{\}$
4: **for** $t = 1, \ldots, T$ **do**
5: $A \Leftarrow D_B(b_i, h_j), b_i \in B'_t, h_j \in H_{open}$
6: $temp \Leftarrow \{\}$
7: **for** $(b_i, h_j, d) \in Hungarian(A)$ **do**
8: **if** $d < \mu$ **then**
9: Add b_i to h_j
10: **else**
11: Add b_i to $temp$
12: **if** $GapSize(h_j, t) < \gamma$ **then**
13: Add ϵ_t to h_j
14: **else**
15: Strip any trailing ϵ and move h_j from H_{open} to H_{closed}
16: **end if**
17: **end if**
18: **end for**
19: Extend H_{open} with $temp$ as new tracks
20: **end for**
21: Extend H_{closed} with H_{open}
22: **return** $\{h \in H_{closed} | \|h\| > 1\}$

Although by definition the Hungarian algorithm performs an *exhaustive* assignment, it is not required that the number of detections B_t equal the number of hypothesis H_t. Hence after processing frame t, some detections may not have been assigned to a hypothesis. This may for example be the case when a new person enters the scene. We thus use these detections to initialize new hypotheses in H_{t+1}, used for processing the next frame. This is also performed when a detection was in fact assigned, but its assignment cost is simply too high (see Algorithm 1 lines 10, 11 and 19).

2.3. Track refinement

Coarse tracks contain many gaps. Writing $h = [\ldots, b_a, \epsilon_{t_1}, \ldots, \epsilon_{t_n}, b_z, \ldots]$ for a coarse track in G, with a gap $[\epsilon_{t_1}, \ldots, \epsilon_{t_n}]$, we call b_a and b_z the spanning vertices. Cause to the gap are the strict matching constraints in coarse tracking. Hence the empty vertices were preferred over other detections. At this stage however, using the spanning vertices as a context, we relax the matching constraints and replace occurrences of ϵ by proper detections whenever possible. Furthermore we decrease the likelihood threshold to include also the more difficult detections. This is what we have denoted as track refinement, and pseudocode is shown in Algorithm 2.

Algorithm 2. Track refinement

Require: $B = \{b_1, \ldots, b_N\}$
Require: $G = (V, E)$ {Cost of edges defined by Eq. (7)}
Require: $\theta = [\theta_0, \theta_1, \ldots, 0]$
$H \Leftarrow CoarseTracking(B, \theta_0)$
for $\theta_i \in [\theta_1, \ldots, 0]$ **do**
$V' = \{b \in B | L(b) > \theta_i\} \cup \epsilon$
$G' = (V', E)$
for $h \in H$ **do**
for every gap $[b_a, \epsilon_{t_1}, \ldots, \epsilon_{t_n}, b_z]$ in h **do**
$p \Leftarrow A^*(b_a, b_z, G')$
Replace $[b_a, \epsilon_{t_1}, \ldots, \epsilon_{t_n}, b_z]$ in h with p
end for
end for
end for
return H

So, in track refinement we iteratively search for alternative (better) paths between each pair of spanning vertices. That is, we perform refinement with hysteresis: At each iteration we decrease the likelihood threshold, $\theta_0 > \theta_1 > \theta_2 > \cdots$ (where θ_0 was used for coarse tracking), and search for a path that includes more detections and less empty vertices. This is performed as a directed graph search (in our case A^*) in G.

2.3.1. Cost and heuristic

A^* depends on two measures; the cost and the heuristic function. The cost should somehow reflect the difference between two vertices. The nature of the graph search, which is a directed search and thus an interpolation, allows us to use more measures in addition to the Bhattacharyya measure that we used in coarse tracking. The Euclidean distance in the xy-position between detections, for example, was unstable for coarse tracking because of possible panning and tilting of the camera. Now that we have a start and goal detection however, the Euclidean distance provides a good grip on how near the person is to its final (relative to the search) position. Similarly we now measure the size of detections, which would previously have led to difficulties e.g. when the camera zooms.

The sum of the three measures could form a good cost function if their values would be somehow aligned. However, although the Bhattacharyya distance is by definition bound between 0 and 1, the upperbound on Euclidean distance and size difference is difficult to define. We decided to divide each of the distances by their sample mean, which is determined in a separate training set of annotated data. This leads to the following cost between two vertices b_i and b_j

$$C(b_i, b_j) = \frac{D_B(b_i, b_j)}{\overline{D_B}} + \frac{D_E(b_i, b_j)}{\overline{D_E}} + \frac{D_{size}(b_i, b_j)}{\overline{D_{size}}} \tag{7}$$

where D_B is the Bhattacharyya distance as in the previous section, D_E is the Euclidean distance between the two detection centers, and D_{size} is the difference in size.

The size is here calculated as the square root of the surface area. The distance measures are normalized with respect to the independent training data, and are thus decorrelated.

We notice that C is a metric and hence the cost between the spanning nodes $C(b_i, b_j)$ is the absolute minimal cost. That is,

$$\forall b_k : C(b_i, b_j) \leqslant C(b_i, b_k) + C(b_k, b_j) \tag{8}$$

By definition this means the cost function is admissible, which makes it an ideal heuristic as well.

2.3.2. Cost to ϵ

As defined by A^* the path cost (between two spanning vertices) is the summed cost over all its edges:

$$C([b_a, \epsilon_{t_1}, \ldots, \epsilon_{t_n}, b_z]) = C(b_a, \epsilon_{t_1}) + C(\epsilon_{t_1}, \epsilon_{t_2}) + \cdots + C(\epsilon_{t_n}, b_z) \tag{9}$$

As we noticed in the previous section, the absolute minimal cost for this path is $C(b_a, b_z)$, and thus an average cost of $1/(n+2)\, C(b_a, b_z)$ at each edge. Knowing this value allows us to control the variation in the appearance of target objects. It allows us to control a preference between smooth tracks with large gaps and noisy tracks with almost no gaps. We express this preference in the parameter α.

$$C(\cdot,\epsilon) = \frac{\alpha}{n+2} C(b_a, b_z) \quad \alpha > 1, \quad b_a, b_z \text{ spanning vertices.} \tag{10}$$

Setting α close to one causes the algorithm to select only the very best matching detections, and more often than not to pick the empty vertex as an alternative instead. Conversely setting α very high (in the order of 3 or above) will cause the algorithm to prefer selecting dissimilar detections over gaps in the track. So in general higher values of α produce further refined, but also more noisy, tracks. An adequate value for α depends on the similarity between different targets.

2.4. Summary

The tracking algorithm we proposed in this section focuses on matching detections over large temporal volumes. To efficiently search in such large sets of detections we represent them in a graph and use a heuristic search to find optimal paths. We find an initial set of coarse tracks which are then refined with hysteresis, slowly loosening the initially strong matching constraints. The parameter of primary interest here is θ which controls that pace.

Track refinement revisits all skipped frames and based on a heuristic makes a decision to leave or replace each empty vertex. Because empty vertices have no appearance, the cost of selecting such a node is calculated as a fraction of the absolute minimal cost between their spanning vertices. This fraction is determined in the parameter α. Higher values of α put a preference on selecting actual detections and thus produce further refined tracks.

3. Tracking complexity

In the previous section we described how we perform tracking in the discretized space of person detections. In this section we analyze the approach from a resources perspective and examine the computational complexity in both time and memory.

The coarse tracking algorithm is efficient in that it iterates only once over the complete sequence of frames. At each frame it assigns the new detections to the existing hypotheses (Algorithm 1 lines 8 and 9), and creates new hypotheses from the unassigned detections (lines 11 and 19). As the assignment at each frame is solved in $O(n^3)$ time [19] with n the number of hypotheses,[1] it is important to keep n low. Taking m as the average number of detections per frame, then after t frames, maximally mt hypotheses have been added. This can only be the case when in Algorithm 1 the condition in line 8 is never met, and thus no assignment was made. Seemingly the number of hypotheses is unbounded. However, a boundary is posed by the maximum allowed gap size, which closes hypotheses that have not been assigned a detection over γ frames as shown in lines 14 and 15. Thus, in the worst case no hypothesis is assigned a detection before exactly γ frames. This means that all of the available hypotheses must have one detection in γ frames. With m detections per frame, there are thus maximally γm simultaneous hypotheses, which makes the time complexity for the assignment polynomial in the number of detections and the maximum allowed gap; $O((\gamma m)^3)$.

In terms of memory the coarse tracking algorithm is comprised of two parts. For one, it has all open hypotheses in memory. In the worst case the hypotheses cover all detections throughout the entire video, in which case we need to store ml detections where l is the length of the video. Exploiting the fact that coarse tracking uses only the last detection in each hypothesis though (see Algorithm 1 line 5), we could reduce the memory burden to not exceed γm detections (plus some references to elements of ϵ which can be ignored as it is only fictitious). Second, the Hungarian algorithm takes roughly $(\gamma m)^2$ for the assignment matrix plus some pointers. Added together, the coarse tracking memory footprint is $O((\gamma m)^2 + (\gamma m))$.

Overall we can say that coarse tracking is very efficient. In fact, most computation is spent refining the tracks, primarily because that is repeated multiple times.

Track refinement finds more detections to each hypothesis by examining its gaps one by one (see Algorithm 2 lines 6–9). A path is searched from one detection to a later detection through a graph of intermediate detections. The complexity of a graph search is generally determined by the branching factor, m, and the depth of the solution, γ. In general this would yield a search space of size m^γ, *i.e.*, if we were to search a tree. However, because the graph is k-partite, the size (in terms of vertices) is linear in the number of frames. Adding the empty vertex to the average branching factor this yields a search complexity of $\gamma(m+1)$.

In the worst case, A^* thus expands $\gamma(m+1)$ nodes. For this behavior, A^* must track back m times in each frame. Backtracking is only performed if, after expanding a node, each of the new nodes have an f value that is worse than that of previous nodes. Noticing that the empty vertex is relatively cheap to pick, backtracking over all nodes only happens when all m nodes are good candidates. As the heuristic measures the color histogram together with the xy position and scale (see Eq. 7), this means that the good candidates are all similar looking and at the same location in the scene. In other words, they are probably multiple detections of the same person. This is not likely to happen over each of the γ frames, especially because the person detection algorithm automatically eliminates largely overlapping candidates. So in practice the worst case will never appear.

Finally, coarse tracking maintains at most γm simultaneous tracks, with each having at least one detection in $\gamma+1$ frames. Therefore in the worst case $\frac{\gamma lm}{\gamma+1}$ γ-length gaps are refined, taking $(m+1)p$ steps to compute each. Track refinement time complexity is thus linear in the length of the video and the maximum allowed gap. It is quadratic in the number of detections per frame.

On a practical note, on each of the videos in the datasets tracking cost well under a second. Setting the maximum gap to more than 300 frames we have been able to find the algorithm take about three seconds. This clearly shows that in practice the algorithm has virtually no constraints and consequently a detailed analysis of practical runtime and memory usage, as opposed to the theoretical worst case, was left out of this paper.

Overall we conclude that, although a video may contain a large amount of detections, person tracking is performed fast and efficient, amongst others due to the structure of the graph, G, that is exploited in our algorithm. Viewed from a calculation perspective, the analysis performed by our multiple person tracking algorithm can save much time in reviewing large amounts of video. As shown, the complexity of the whole tracking algorithm is polynomial in time and memory. Most computation depends on the maximum allowed gap size and the number of detections per frame.

4. Experiments

In this section we present an evaluation of our tracking algorithm. Section 4.1 regards our dataset and Section 4.2 the evaluation methodology. Experimental results on our dataset are presented in Section 4.3 and on the PETS2009 dataset in Section 4.4.

4.1. Dataset

The video data comprises 32 separate one-hour recordings taken from real-life police operated surveillance cameras in the

[1] Actually n is the maximum of the number of hypotheses and the number of detections, though generally the first is much larger.

Fig. 2. Sample frames. A few frames to show the diversity in real-life video.

Amsterdam city center. Each of the videos shows vast amounts of people walking around plus loads of other activity. A few examples are given in Fig. 2. The advantage of such recordings is that the evaluation is very realistic, not only in the sense of how people interact with the environment, but also in the sense of surveillance video characteristics like moderate image resolution and extremely low and variable frame rates. In fact regulations dictate that at least three frames are recorded every two seconds [12].

At the time of recording ten actors walked different routes through the city center. On average five actors appear in each video. Aside from the actors there are many other people present; roughly 700 per video, with an average of two people visible per frame. Not occasionally, however, more than five people are visible simultaneously, both in groups and as individuals.

The visual quality of the video frames, although with a resolution of 720×576 pixels, is quite degraded showing large compression artifacts and signal noise (unstable pixel values). The frame rate is variable, with an average of four frames per second. In the scene, shadow, reflections, water and trees cause fair amounts of noise. The lighting fluctuates rapidly and in some cases also the overall chroma changes significantly.

Next to our own dataset, we also run the algorithm on a publicly available benchmark pedestrian dataset, PETS2009 [10]. The S2 part of the dataset is aimed at pedestrian tracking and contains seven different recordings of a crowd in a relatively clean scene without other traffic. Each video fragment consists of roughly 800 frames of 768×576 pixels each, shot with a static camera. At each time instant there are about ten people in the scene simultaneously. Unfortunately the dataset's ground truth labels are not publicly available and consequently we are unable to evaluate our tracker on this data in the same way we do it on our own dataset.

4.1.1. Annotation

Because of the huge amount of persons in the video data, we restricted ourselves to annotating only the actors. Each annotation marks an entire person track as a sequence of bounding boxes tightly fit around the person's contour. Person images that are more than half occluded by objects in the scene (*i.e.* non-persons) are not counted and thus the track is interrupted. If this interruption is less than the maximum allowed gap, γ (as introduced in Section 3), the track is still continued after the gap. In total this led to 240 tracks comprised of 30,772 bounding boxes.

Because we felt that the annotation of only the actors is not sufficient for a good evaluation of the tracking algorithm, we also fully annotated a five minute video fragment showing one of our actors plus 49 other people. This resulted in the annotation of 56 tracks (some people reappear in the scene at a later moment), and comprises 2896 bounding boxes. The fact that tracks in this fragment are much shorter than in other videos is due to a slightly zoomed camera and the scene, which contains large occluding objects.

4.2. Evaluation

A natural way of evaluating tracks is to count the overlaps between detected tracks and annotated tracks. This works well when for each detected track it is known which object it is tracking, for example when initial object locations are assigned by a user. In our case however, tracking is performed fully autonomously and there is no explicit correspondence between hypotheses and annotated tracks. As a consequence we use different measures to assess the tracking performance.

First we evaluate each stage in our tracking algorithm separately. We quickly evaluate our input, the person detections resulting from Dalal and Triggs [9], and then its impact on coarse tracking, followed by an evaluation of track refinement. We conclude with the multiple object tracking (MOT) performance metric as proposed by Bernardin and Stiefelhagen [5]. Before we commence on the actual evaluation though, we first put forward some simple definitions that are used throughout the evaluation.

4.2.1. The annotation

Evaluation is performed by matching detections to the manually annotated ground-truth. Writing O for the complete set of annotated person tracks,

$$o = [l_1, \ldots, l_n]; \qquad o \in O \tag{11}$$

is a single person track consisting of n annotations. From this point onwards we will call annotated person tracks *objects*. Just as with the hypotheses an object can hold no more than one annotation per frame.

To determine if an annotation, l, is matched by a detection, b, we calculate the ratio of the area of overlap with respect to the individual areas. A threshold determines if the overlap is sufficient to accept it as a match.

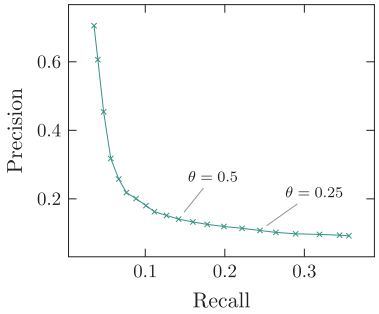

Fig. 3. Detection evaluation. Precision-recall curve for the used person detection algorithm.

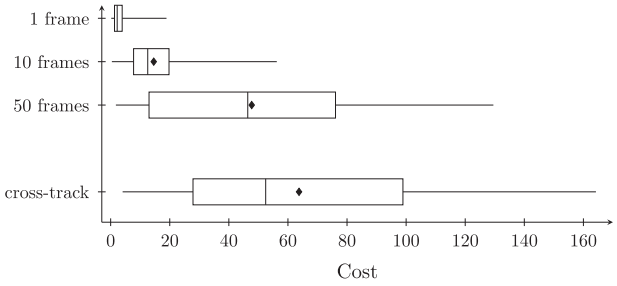

Fig. 4. Cost between detections. The box and whisker plots show the variation of cost, $C(\cdot)$, between successive detections. The first three show variation *within* labeled tracks over 1, 10, and 50 frames respectively. The last plot shows the variation between detections of *different* tracks. In this latter case the gap size has no noticeable effect and so only that for a 10 frame gap is shown.

$$P(r_1, r_2) = \begin{cases} 1 & \text{if } \frac{A(r_1 \cap r_2)}{\frac{1}{2}(A(r_1)+A(r_2))} > \tau \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

where $A(\cdot)$ denotes the surface area of its argument. The fraction divides the overlapping surface by the average surface area of the two polygons. Whereas a threshold of $\tau = 0$ accepts any two touching regions, $\tau = 1$ would only accept identical regions in shape, size, and location. We take $\tau = 0.6$ as a reasonable evaluation, accepting only regions that clearly mark the same area and thus the same object.

This function is easily extended to compare whole tracks:

$$Q(h, o) = \sum_t P(h_t, o_t) \quad (13)$$

counting the number of matched annotations in o (or equally, the number of matching detections in h), where $P(h_t, o_t)$ is taken to be 0 when either h or o has no polygon at time t.

4.2.2. Recall of annotations and objects

As indicated before, we use the person detection algorithm from Dalal and Triggs [9]. With a posterior threshold on the detection likelihood from Eq. 3, we calculate the precision and recall averaged over all frames. The curve is shown in Fig. 3. It confirms that the confidence of correctly detecting persons (precision) increases with $L(b)$, and similarly the recall decreases. Please notice that the precision and recall are measured here over the individual detections, not regarding tracks.

On a final note of evaluating the detections, we can evaluate the recall of the individual detections with respect to the annotated *objects*. Although a measure for precision cannot be expressed this way, we can calculate the number of tracks that are matched by detections. Since tracks are only built from these detections, this measure poses an upper limit for the tracking performance, and hence is crucial for the interpretation of tracking results in the following sections.

We define an object in the ground truth to be detected when at least two of its annotations are matched. The number of detected objects with respect to the total number of objects can be regarded as a sort of recall with respect to the annotated tracks, O.

To evaluate the tracking algorithm we need more detailed measurements than simply calculating the precision and recall mea-

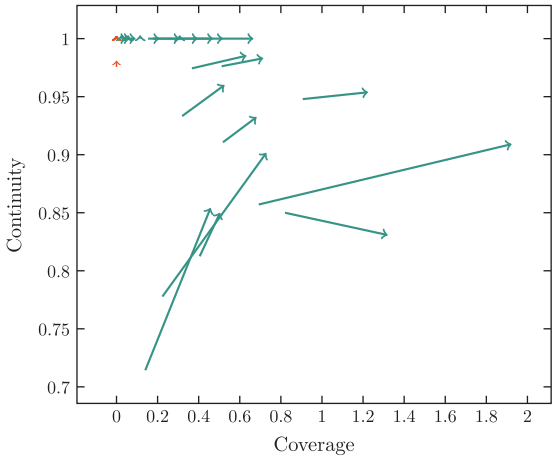

Fig. 6. Tracking progress. Each line represents a track. The direction of the line shows the performance change over iterations of track refinement. Because non-person tracks have no coverage value defined, we set it to 0.

Table 1
MOT metrics. We evaluate both on *all* generated hypotheses H, and on the subset of hypotheses that—after refinement—mark a person in at least one detection. Recalling Eq. (13) that is the subset of hypotheses for which $\exists o$: $Q(h, o) > 0$. As this selection by definition only affects $\overline{fp}$, and consequently MOTA, the other values have been printed in a gray tint.

	MOTP	MOTA	$\bar{m}$ (%)	$\overline{fp}$ (%)	$\overline{mme}$ (%)
Coarse	0.7821	−0.5038	76.7	72.9	0.8
Refine	0.7464	−0.6084	67.5	92.4	0.9
Coarse, $Q > 0$	0.7821	0.1796	76.7	4.6	0.8
Refine, $Q > 0$	0.7464	0.2186	67.5	9.7	0.9

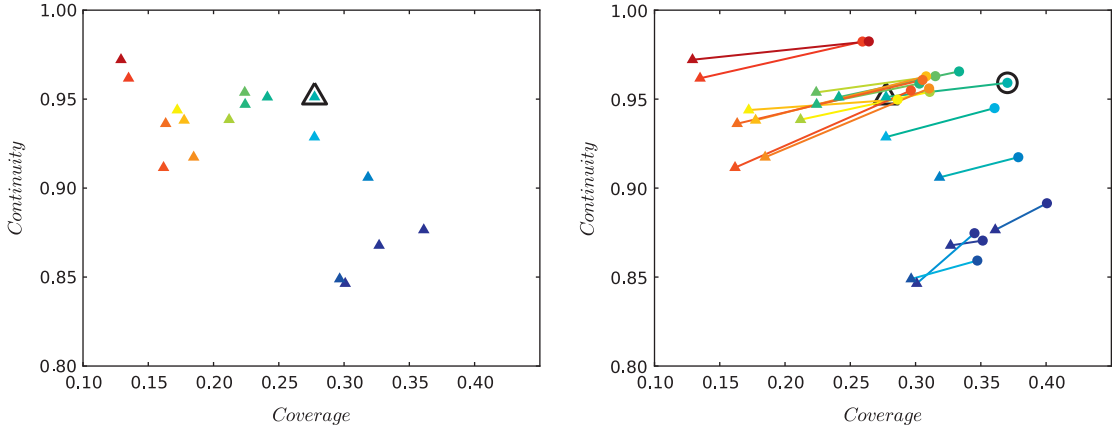

Fig. 5. Choosing θ. The *left figure* shows the continuity-coverage values for coarse tracking at different parameter values (top left $\theta_0 = 1$, bottom right $\theta_0 = 0.05$). Each point averages the scores over all tracks in the dataset. The *right figure* extends on the first and shows the performance increase resulting from track refinement. Each circle marks the continuity-coverage values at $\theta_n = 0$. The highlighted points mark $\theta_0 = 0.5$.

Table 2
Comparing performance. We compare the overall performance of our tracking algorithm against the algorithms described in Section 4.2.5. Our algorithm outperforms the OpenCV multiple object trackers on both MOTP (remember lower MOTP values are better) and MOTA even though the best individual miss rate is achieved by OpenCV23MTTa and the best mismatch rate by OpenCV23MTT. The odd value of 111.2% false positive rate is actually correct (see the final remark in Section 4.2.4).

	MOTP	MOTA	$\bar{m}$ (%)	$\overline{fp}$ (%)	$\overline{mme}$ (%)
Refine	**0.2724**	**0.1198**	73.7	**12.2**	2.2
OpenCV23MTT	0.7759	−0.6557	53.9	111.2	**0.5**
OpenCV23MTTa	0.7743	−0.1523	**37.4**	75.9	1.9

Bold values mark the best performance.

sured over numbers of tracks. For example because a hypothesis can match multiple objects and because objects can be matched by multiple hypotheses. Therefore in the following section we introduce a more thorough evaluation measure of multiple target tracking algorithms.

4.2.3. Quality of tracks

One characteristic of good tracks is that they follow one object (person) *continuously*, *i.e.*, not accidentally matching other objects. We calculate the continuity by comparing each detection in a track to its next detection. An error is counted when the detections do not mark the same person. The track continuity is expressed as the number of continuous detections with respect to the total track length as follows:

$$F(h) = \frac{|\{h_i \in h \mid h_i = h_{i+1}\}|}{|h| - 1} \tag{14}$$

resulting in a continuity score between 1 for monotonous tracks of exactly one object and 0 for completely discontinuous tracks.

Continuity measures the tracking performance within the hypothesis, but it does not tell how much of the original object was covered. To that end we introduce the *coverage*. It divides the track length of the hypothesis by the track length of the object that it matches best.

$$G(h) = \frac{Q(h,o)}{|o|}, \quad o = \arg\max_{o \in O} Q(h,o) \tag{15}$$

This leads to a value between 1 for a fully matched annotation and a very low value (not zero because obviously the annotation has a positive length) for matching only a fraction of the annotation.

Optimally a hypothesis would fully cover an object without matching any other annotation. In that case both continuity and coverage will have a value of 1. If, however, the object is fully matched but another object is also sometimes matched, we expect the continuity to drop while the coverage stays at 1. Conversely, when all detections match the single object but the hypothesis fails to match *all* of its annotations, the continuity is 1 while coverage is very low. This shows that the continuity and coverage conveniently describe two complementary object tracking qualities.

4.2.4. Multiple target tracking

The accuracy and coverage metrics proposed in this paper illustrate the performance effects of coarse tracking and refinement specifically. For a more general evaluation of multiple object tracking certainly more evaluation metrics exist [26,20,5]. Particular to the approach by Bernardin and Stiefelhagen [5] is that they try to provide a "human" value of judgment. This is achieved by accepting only one hypothesis per object and by rejecting a hypothesis not before it truly fails to match the object. In other words, alternative hypotheses may – on a per-frame basis – match the object better, but they do not contribute to refuting the first hypothesis.

Using this linking strategy, the method employs two performance measures, MOTP and MOTA. The first measures the localization error measured as the average distance between each detection in a hypothesis and its corresponding annotation in the object. We calculate the distance as one minus the overlap ratio, which in turn is calculated as the surface area of the overlapping region divided by the sum of surface areas (see also part of Eq.

Fig. 7. A simple example. The person contrasts well with the rest of the frame and is thus easily tracked.

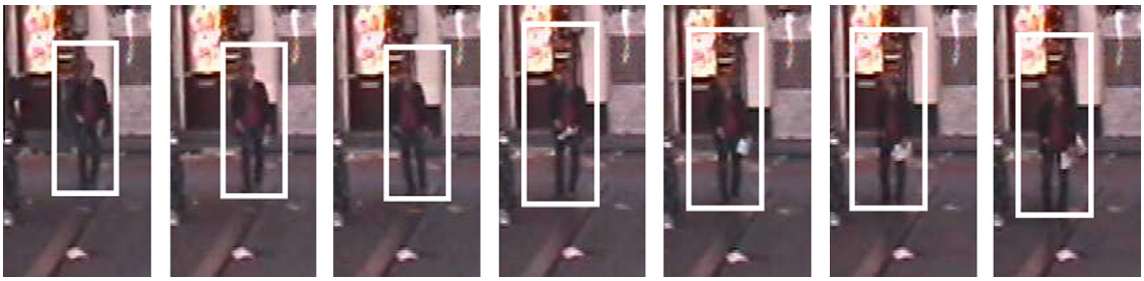

Fig. 8. Noisy images. Noise in the scene and noise from image compression reduce detection quality and likelihood values – here not all bounding boxes are placed accurately. Still, with only two detections from coarse tracking (the two outermost images) the rest can be recovered with hysteresis.

Fig. 9. Occlusion. Two examples of occlusion. The person marked in black walks in front of the person marked in white. Using coarse tracks the gap is automatically skipped. Refinement adds a few detections to narrow the gap afterwards.

Fig. 10. Selecting poorer detections. As seen in this example refinement selects detections of inferior quality. Still the detections may be very useful.

Fig. 11. Refinement errors. Two examples of mistakes in track refinement. The left and right most images are taken from the coarse track while images in the middle were added by refinement. In the first example the correct person was simply not detected with a satisfactory likelihood causing refinement to prefer a nearby alternative. In the second example the spanning detections were already discontinuous. Refinement as interpolation finds intermediate detections similar to both endpoints, which causes these alternations.

12). The second measure, MOTA, expresses an average of three different error rates—the miss rate $\bar{m}$, mismatch error rate $\overline{mme}$, and false positive per object rate $\overline{fp}$. Each measure relates their error to the number of objects that are present in the video. A *miss* counts when an object is not detected, a *mismatch* means the hypothesis for an object is switched, and the *false positives* count the surplus

Fig. 12. Walking side by side. Two people walk side by side. The tracks are kept separate.

Fig. 13. False positives. The space of detections B consists of mainly false positives. As a consequence these detections are tracked as well. There is little interference though with the real person tracks and thus they are easily removed in a post-processing step.

of hypotheses. Notice that the false positive rate in this calculation differs from the generally accepted false positive rate. Where the *regular* false positive rate divides by the number of hypotheses, resulting in a value between 0 and 1, the false positive *per object* rate divides by the total number of objects which consequently leads to a value between 0 and infinity.

4.2.5. Comparison to other work

We compare our results against the publicly available multiple object tracker shipped with OpenCV 2.3 [1]. The code uses connected components for blob entrance detection, mean shift particle filtering for blob tracking, and Kalman filtering for blob trajectory post processing. We will label this tracker *OpenCV23MTT*. Comparing against this tracker will highlight the performance of our tracking algorithm as a whole – including the person detection module.

To evaluate the performance of our coarse tracking + refinement procedure regardless of the underlying person detection, we slightly modified the OpenCV tracker by having the blob entrance detection read the same bounding boxes from the person detection module. Here, both our tracking algorithm and the OpenCV tracker have the same basis of detected objects and therefore any performance difference must be purely due to the tracking mechanism rather than the detection. We will label this tracker *OpenCV23MTTa*.

4.3. Results

To give us an idea of the cost function, as defined in Eq. 7, the box plot in Fig. 4 shows the cost between detections of a single person and the cost between detections of different people. On consecutive frames the top 25% within-track cost values start to overlap with the smallest between-track cost values. Ten frames apart the detections still seem to be reasonably separable.

Following the suggested evaluation in Section 4.2.2, we calculate the number of objects that are detected in the person detection algorithm for varying thresholds. At $\theta_0 = 0$, so using the full set of detections B, two thirds of the tracks are hit. This means that one in three persons cannot be tracked at all simply because it is not detected in any of the frames, a discussion of which we will come to in the next section. At $\theta_0 = 0.5$ we have the subset of detections $\{b \in B | L(b) > 0.5\}$. This subset omits a few more tracks, *i.e.* 50% of the tracks are still hit. This performance is maintained in coarse tracking which does indeed find all of these tracks. Thus, setting $\theta_0 = 0.5$ results in a set of coarse hypotheses matching half of the annotated objects – counted in the number of tracks.

We will come to an evaluation of different threshold values below, but for now we accept a recall of 50% of the tracks and perform coarse tracking at $\theta_0 = 0.5$. At this level the matching hypotheses hold 23.8 detections on average. When we refine the tracks, decreasing θ to 0 in steps of 0.1, the algorithm overall adds 13.7 more detections per track, roughly 57%. This is reflected in the increased coverage as shown in Fig. 5bb. At the same time the figure

Fig. 14. Large scale change. Two examples of perspective. Both people walk away from the camera causing their scale (as measured in pixels) to decrease tremendously. Still we are able to track them. Also notice that in the first example the person is taking his bag to his front, hiding it from the camera.

shows that track refinement increases the continuity. In Fact 83% of the added detections is correct.

We now evaluate possible values for θ_0. Fig. 5aa shows a decrease in coarse tracking coverage when θ_0 is increased. Thus the tracks contain larger gaps. At the same time however, the continuity increases with θ_0. So although the tracks contain fewer detections, they are qualitatively better, which makes sense because particularly correct person detections tend to have high likelihood scores. The choice for this threshold thus poses a trade-off on the tracks: either they contain smaller gaps but the tracks are less continuous, or the tracks are more continuous but contain larger gaps to be refined. Fig. 5bb shows that refinement is particularly capable of refining large gaps, *i.e.*, increase in coverage is larger at higher values for θ_0. The optimal value for θ_0 does indeed lie at 0.5, where the performance on average lies closest to the optimum continuity and coverage which is at $(1,1)$.

In Fig. 6 we have chosen $\theta = [0.5, 0.4, 0.3, \ldots, 0]$ and zoom in on the performance of individual tracks. The figure shows that most tracks are actually fully continuous to start with. For those tracks their coverage increases and their continuity stays at 1 which means that every added detection from refinement is correct. In some cases though, we do notice a decreased continuity. In these cases the coarse tracks already contain some discontinuities that are propagated and sometimes worsened by track refinement. An often seen scenario is that a gap between a detection of person A and a detection of person B (thus being discontinuous already in the coarse track) is filled with alternating detections of person A and B, which pulls down the continuity even further. An example of this is given in Fig. 11.

Table 1 shows the tracking performance in terms of the MOT metrics. We notice three particular properties of the tracker. First we notice the MOTP, the average overlap ratio of successful detections, which is on average 0.78 in coarse tracks. This means that the overlap between a detection and annotation of the person is on average 0.22. Refinement decreases this value slightly, and thus increases the performance, probably because it has the coarse track as a context and is better capable of finding the right detections. This proves the relevance of first having a coarse tracking stage. A clear example of this is shown in Fig. 10.

Secondly we see that by far the most false positives are found in hypotheses that do not mark a person *at all*. This is because of two effects. For one, the dataset was only annotated with the actors, so detected non-actors will be counted as false positive. Secondly, the person detection algorithm we used simply generates an overwhelming number of false detections. However, as is shown in Fig. 13, these detections are all very well "tracked" too, in the sense that parts of the scene that often cause false positives are collected together in separate tracks. With a post-processing step these non-person tracks can easily be discarded at once.

Then the third property we regard is the miss rate $\bar{m}$ which is decreased almost 10% by track refinement. Even stronger, after removing the non-person tracks we see that the performance gain from including missed detections does indeed outweigh the increased false positive rate. Consequently we see the MOTA rise. Refinement thus not only adds to the number of detections per track, but at the same time increases the accuracy.

Table 2 contrasts the performance of our algorithm with the other two multiple object trackers. The evaluation is carried out on the fully annotated video fragment in the Amsterdam dataset. The numbers illustrate clearly the robustness of our method to time-lapse and low quality video data. The increase in MOTA for OpenCV23MTTa shows that in this kind of video it is better to stick with actual pedestrian detections rather than tracking any "blob" that enters the scene.

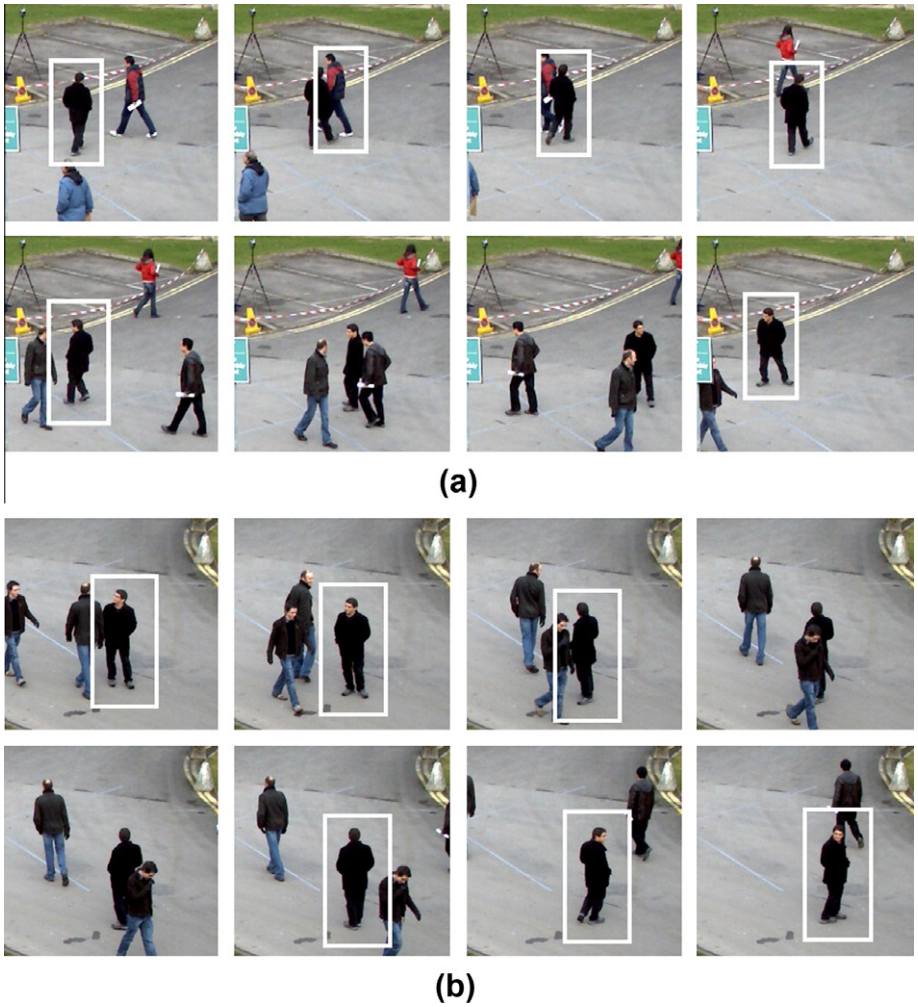

Fig. 15. Track refinement. Two illustrations of detections added by refinement. In both cases the first and last image (in reading order) are taken from the coarse track and the intermediate detections result from refinement. Track (a) is refined from 10 to 43 detections and track (b) from 13 to 30.

Fig. 16. Skipping many frames. After wandering around a bit, the person stops and stands still. The person detection algorithm, which utilizes background subtraction, soon enough looses the person 'out of sight'. After more than 60 frames the person starts walking again, he is detected and the track resumes.

We show some examples of tracks in Fig. 7–14, with different figures highlighting different aspects about the tracks. Each row of images shows a selection of detections. The cut-outs are taken all at the same pixel size to provide clear insight in scale changes. Most sequences are also cut out at the exact same position within the frame to provide an understanding of people's motions. In some cases multiple tracks are jointly shown to illustrate the multiple person aspect of tracking. In these cases the different

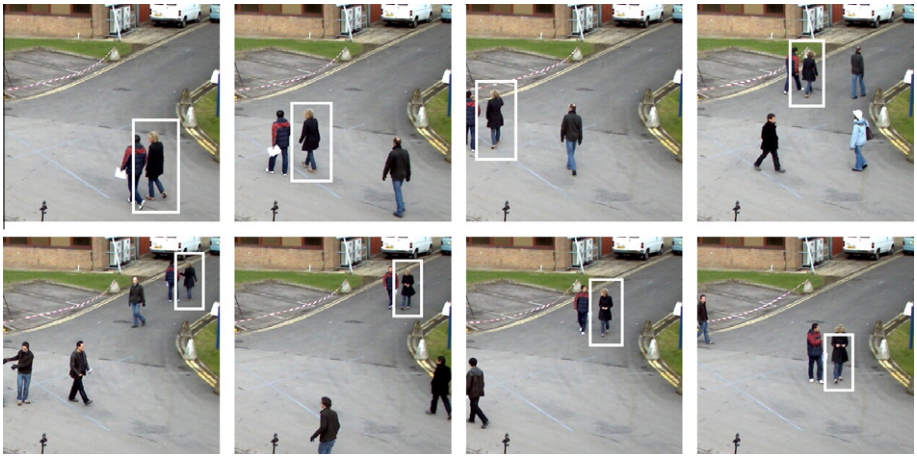

Fig. 17. Scale. A track over 313 frames with a change in scale due to perspective. Furthest away the person appears in the image three times smaller than when closest to the camera.

tracks are annotated in contrasting tints, *i.e.*, black bounding boxes mark one track, white boxes the other. Particular attention should be drawn to the examples in Fig. 9, showing how coarse tracks are a very powerful way to deal with occlusions and multiple persons. In both the examples the occluded person's track shows a gap for the duration of the occlusion and successfully continues thereafter.

4.4. PETS

In this section we present results obtained from running the presented algorithm on the public benchmark pedestrian dataset PETS2009. Because this dataset does not provide the ground truth labels, rather than presenting a numerical evaluation, we instead show some fragments of generated tracks for analysis.

In comparison with tracking on our own dataset, we notice that the precision of the person detection seems to be higher. This is probably because the quality of the images is much higher and less effected by sunlight, dust, reflections and other noise from the scene. This has also a bearing on the amount of detections present in coarse tracking, which is a bit higher because more detections have a high likelihood score.

Similar to the case on our dataset the improvement on the miss rate after refinement is much higher than the increase in false positive rate. Examples of this are shown in Fig. 15. In the first example refinement adds 33 more to the 10 detections in the coarse track. Two of those are false positives (one shown in its second image). In the second example the number of detections increases from 13 to 30 without any false positive, even though the occluding person looks quite similar.

The distinct advantage of our tracking algorithm—being able to skip many frames—is clearly illustrated in Fig. 16 where the track is continued after the person has been 'out of sight' for over 60 frames. Because here the person is standing still for a long time, its pixels are gradually incorporated into the background model. As soon as the person starts moving again, he is detected and tracking continues. The gap cannot be refined however, because detections of the person are truly missing.

Finally we show a nice example of the tracker being able to cope with scale change of the target objects. The track shown in Fig. 17 follows a person walking away from the camera, turning around, and walking back to the camera. Due to perspective the size of the person in the frames changes roughly threefold. The track is more than 300 frames long with only two misses.

5. Conclusion

As performance of object detection algorithms in static images is growing, the fields of object detection and object tracking are coming together. In fact, in this paper we introduced a tracking algorithm that is entirely based on such object detections. This approach has two clear advantages. First, by building a graph from the detections we allow our algorithm to 'backtrack' over many frames. As such it is guaranteed to find globally optimal paths. Secondly, because of this structure we can create paths that skip many frames, and this is actually meaningful! Often, either because of occlusion or camera noise or other circumstances, the uncertainty is too large to continue a track. Whereas current trackers in such situations blindly choose a continuation that maximizes some criterion, we skip those frames and process them later, when there is more information from surrounding frames.

Evaluation of the algorithm has shown that indeed our approach deals particularly well with occlusion of target objects. When multiple people cross their paths, we see that tracking of occluded persons is postponed until they are again fully visible. Moreover, as refinement is repeated with hysteresis, also the semi-occluded detections are correctly added.

Provided that person detections are available the algorithm processes an hour length video in a matter of seconds. We have analyzed the complexity and have shown that, in contrast with traditional MHT algorithms which scale exponentially, our algorithm scales polynomial in the number of detections. So our method is very efficient and fast.

Regarding the detections however, the detection algorithm we used has shown not to be very accurate with around 90% false positives. One reason might be that the it was trained with images entirely different from our video data. The images used for training were taken from a horizontal angle and show people in high quality and under good lighting conditions. In contrast, our surveillance video data shows people from a birds view perspective, the images show a great deal of compression artifacts, and due to the contrast in sunny and shadowy regions the lighting conditions vary tremendously. Still, despite these real-life video conditions, and despite the amount of false detections, the tracking algorithm has shown good performance keeping non-person and person detections in separate tracks.

On a final note, the presented tracker can be applied on any set of detections, as long as the detections are defined as polygons. It is not limited to persons. For example, using a car detector in traffic video data our backtracker can be used to track cars as well.

Acknowledgments

We would like to thank the Netherlands Forensics Institute and the Amsterdam police force for the opportunity to use real-life surveillance video from cameras in the Amsterdam city center. Additionally, this work was supported by the EC-FP6 VIDI-Video Project and the BSIK MultimediaN Project.

References

[1] Open source computer vision, 2011. <http://opencv.willowgarage.com/> (accessed 23.11.11).
[2] L. Alvarez, J. Weickert, J. Sánchez, Reliable estimation of dense optical flow fields with large displacements, Int. J. Comput. Vision 39 (1) (2000) 41–56.
[3] J. Berclaz, F. Fleuret, P. Fua, Robust people tracking with global trajectory optimization, in: IEEE Conf. on Comput. Vision and Pattern Recognit., vol. 1, 2006, pp. 744–750.
[4] J. Berclaz, F. Fleuret, P. Fua, Multiple object tracking using flow linear programming, in: Proc. Twelfth IEEE Int. Performance Evaluation of Tracking and Surveillance (PETS-Winter) Workshop, 2009, pp. 1–8.
[5] K. Bernardin, R. Stiefelhagen, Evaluating multiple object tracking performance: the clear mot metrics, J. Image Video Process. 10 (2008).
[6] T. Brox, A. Bruhn, N. Papenberg, J. Weickert, High accuracy optical flow estimation based on a theory for warping, in: Eur. Conf. on Comput. Vision, Lecture Notes in Computer Science, vol. 3024/2004, Springer, Berlin/Heidelberg, 2004, pp. 25–36.
[7] C. Chang, R. Ansari, A. Khokhar, Multiple object tracking with kernel particle filter, in: IEEE Conf. on Comput. Vision and Pattern Recognit., vol. 1, 2005, pp. 566–573.
[8] D. Comaniciu, V. Ramesh, P. Meer, Real-time tracking of non-rigid objects using mean shift, in: IEEE Conf. on Comput. Vision and Pattern Recognit., vol. 2, 2000, pp. 142–149.
[9] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: IEEE Conf. on Comput. Vision and Pattern Recognit., vol. 1, 2005, pp. 886–893.
[10] J.M. Ferryman, A. Shahrokni, An overview of the pets2009 challenge, in: Proc. of the Eleventh IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance, Miami, 2009, pp. 25–30.
[11] A. Gijsenij, T. Gevers, J. van de Weijer, Computational color constancy: survey and experiments, IEEE Trans. Image Process. 99 (2011). early Access.
[12] M. Gill, A. Spriggs, Assessing the impact of cctv, Tech. Rep. 292, Home Office Research, Development and Statistics Directorate, 2005.
[13] M. Han, W. Xu, H. Tao, Y. Gong, An algorithm for multiple object trajectory tracking, in: IEEE Conf. on Comput. Vision and Pattern Recognit., vol. 1, 2004, pp. I-864–871.
[14] M. Isard, A. Blake, Condensation—conditional density propagation for visual tracking, Int. J. Comput. Vision 28 (1) (1998) 5–28.
[15] H. Jiang, S. Fels, J.J. Little, A linear programming approach for multiple object tracking, in: Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR '07), 2007, pp. 1–8.
[16] S.M. Khan, M. Shah, Tracking multiple occluding people by localizing on multiple scene planes, IEEE Trans. Pattern Anal. Mach. Intell. 31 (3) (2009) 505–519.
[17] M. Lefébure, L. Cohen, Image registration, optical flow and local rigidity, J. Math. Imag. Vision 14 (2) (2001) 131–147.
[18] M. Metternich, M. Worring, A. Smeulders, Color based tracing in real-life surveillance data, in: Y. Shi (Ed.), Transactions on Data Hiding and Multimedia Security V, Lecture Notes in Computer Science, vol. 6010, Springer, Berlin/Heidelberg, 2010, pp. 18–33.
[19] J. Munkres, Algorithms for the assignment and transportation problems, J. Soc. Indus. Appl. Math. 5 (1) (1957) 32–38.
[20] A.T. Nghiem, F. Bremond, M. Thonnat, V. Valentin, Etiseo, performance evaluation for video surveillance systems, in: IEEE Conf. on Adv. Video and Signal Based Surveillance, 2007, pp. 476–481.
[21] H. Nguyen, A. Smeulders, Robust tracking using foreground-background texture discrimination, Int. J. Comput. Vision 69 (3) (2006) 277–293.
[22] P. Nillius, J. Sullivan, S. Carlsson, Multi-target tracking – linking identities using bayesian network inference, in: Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition, vol. 2, 2006, pp. 2187–2194.
[23] J. Pan, B. Hu, Robust object tracking against template drift, in: IEEE Int. Conf. on Image Proc., vol. 3, 2007, pp. 353–356.
[24] D. Reid, An algorithm for tracking multiple targets, IEEE Trans. Automat. Contr. 24 (6) (1979) 843–854.
[25] K. Shafique, M. Shah, A noniterative greedy algorithm for multiframe point correspondence, IEEE Trans. Pattern Anal. Mach. Intell. 27 (1) (2005) 51–65.
[26] K. Smith, D. Gatica-Perez, J. Odobez, S. Ba, Evaluating multi-object tracking, in: IEEE Conf. on Comput. Vision and Pattern Recognit., vol. 3, 2005, p. 36.
[27] A. Torabi, G.-A. Bilodeau, A multiple hypothesis tracking method with fragmentation handling, in: Proc. Canadian Conf. Computer and Robot Vision (CRV '09), 2009, pp. 8–15.
[28] S. Tran, Z. Lin, D. Harwood, L. Davis, Umd-vdt, an integration of detection and tracking methods for multiple human tracking, in: Multimodal Technologies for Perception of Humans, Lecture Notes in Computer Science, vol. 4625, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 179–190.
[29] B. Wu, V.K. Singh, C.H. Kuo, L. Zhang, S.C. Lee, R. Nevatia, Clear'07 evaluation of usc human tracking system for surveillance videos, in: Multimodal Technologies for Perception of Humans, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 191–196.
[30] C. Yang, R. Duraiswami, L. Davis, Fast multiple object tracking via a hierarchical particle filter, in: IEEE Int. Conf. on Comput. Vision, vol. 1, 2005, pp. 212–219.
[31] M. Yang, Y. Wu, G. Hua, Context-aware visual tracking, IEEE Trans. Pattern Anal. Mach. Intell. 31 (7) (2009) 1195–1209.
[32] L. Zhang, Y. Li, R. Nevatia, Global data association for multi-object tracking using network flows, in: Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR 2008), 2008, pp. 1–8.