Robust Patch-Based Pedestrian Tracking using Monocular Calibrated Cameras

Gustavo Führ, Cláudio Rosito Jung
Institute of Informatics - Universidade Federal do Rio Grande do Sul
{gfuhr, crjung}@inf.ufrgs.br

***Abstract*—Although several methods for pedestrian tracking can be found in the literature, robustly tracking a person in unconstrained environments is an open and active research problem. In this paper, we propose a method that represents each pedestrian as a set of multiple fragments, aiming robustness with respect to occlusions. These patches are tracked individually and their translation vectors are combined robustly in the world coordinate frame using Weighted Vector Median Filters (WVMF). Additionally, the algorithm uses the camera parameters to both estimate the person scale in a straightforward manner and to limit the search region used to track each fragment. Experiments carried out using two publicly available datasets (*PETS* and *TownCentre*) are presented, and they indicate that the proposed method is robust to partial occlusions and large scale changes. According to our experiments, the proposed approach outperforms, regarding the quality of localization, some of the methods in the current state of the art.**

***Keywords*-patch-based tracking; wvmf; homography; pedestrian tracking;**

I. Introduction

Tracking people represents an essential task in a wide range of applications such as automated video surveillance of public areas, video indexing and analysis of athletic performance in collective sports. The task of constantly localizing a pedestrian in a video sequence is a complex one due to several reasons: people can move fast and unpredictably, the subject's appearance can change throughout the sequence and the video can present noise and blur, among others. In a common surveillance scenario, the problem can be even harder because of the many occlusions (between a group of people or the person and scene) that can occur. Additionally, these systems often require on-the-fly execution, so non-causal methods that use future information to estimate the current state are not well suited for these applications.

Methods based on multiple fragments [1], [2] have been shown to increase robustness in the presence of partial occlusions. In this paper, we extend the method proposed by Dihl et al. [2] to the specific case of pedestrian tracking using a single calibrated camera. Our framework is independent to the image features that are used, i.e. different kinds of features can be easily accommodated in the proposed technique. More specifically, our contributions with respect to [2] are three-fold:

- A method that is able to cope with partial occlusions by treating the target as a set of independent parts, called fragments, and combining their motion in the World Coordinate System (WCS).
- A simple operation that uses camera parameters to accurately estimate the change in scale of the target.
- An adaptive procedure that is able to define the search region at each frame, which is computed using the position of the person in the world.

The remainder of this paper is organized as follows. Section II presents some related work on pedestrian tracking, and Section III describes the proposed approach. Experimental results and a discussion of the proposed method are provided in Section IV. Finally, we draw our conclusions in Section V.

II. Related work

Several methods for region-based object tracking can be found in the literature (see [3] for a complete survey on the problem). A classical approach is template matching, which works by minimizing some kind of error function (e.g. sum of squared differences) between the template and the current frame intensities at several candidate locations. These methods work well if the target presents mostly a translational motion in the image and if its appearance does not change too much over time. Since a person can appear in a variety of poses and the subject appearance can change over time significantly (especially in a monocular scenario), this class of methods are not well suited for pedestrian tracking.

Different image features can be used for pedestrian tracking. Many algorithms rely on background segmentation to detect and track people. For instance, the W4 [4] algorithm uses background segmentation, shape and texture information to perform real time tracking in a monocular gray-scale video. One drawback of techniques that rely on background removal is that the camera is usually assumed to be static. Color information can also be employed to model people appearance and can significantly increase tracking performance. Ramanan and Forsyth [5] proposed a method to detect and track people using a 2D body model and an appearance model constructed by finding a mean color histogram of different feature candidates extracted in the first few frames of the video. In [6], the intersection between a foreground mask and the frame is used to select the pixels that compose the subject appearance, modeled in a probabilistic framework.

In addition to the choice of features, a challenging aspect of pedestrian tracking is to maintain a good localization during and after an occlusion. This is critical for surveillance systems, in which partial and total occlusions can happen very often. One way to approach this issue is to use several cameras to reduce the ambiguity induced by an occlusion. The method proposed by Fleuret et al. [6] models the problem of multi-view people tracking as an inference in a Hidden Markov Model. Although methods in this category present some promising results, multi-camera setups are not always an option for most surveillance systems. In recent years, methods based on data association were proposed for multiple people tracking. Benfold and Reid [7] use Histograms of Oriented Gradients [8] (HoGs) and Kanade-Lucas-Tomasi (KLT) tracking to detect people and estimate their motion between detections. To obtain the final trajectories, a Markov-Chain Monte-Carlo data association is applied within a temporal window. One drawback of these methods is the latency caused by the use of future observations in the estimation of the current state, i.e. such approaches are not causal.

Another way to deal with partial occlusions is to consider the target object as a set of adjacent patches. The rationale behind this idea is that if some patches are occluded and tracked incorrectly, the remaining patches can provide a good estimate of the pose. The FragTrack algorithm [1] divides the target region into multiple image fragments at initialization. For each fragment, a vote map is constructed using image histograms. Then, these maps are combined in a robust way so that the influence of outliers is reduced. The work of Dihl et al. [2] also uses the same principal idea for object tracking, but tracks each patch independently. The method includes a motion prediction phase to deal with total occlusions and the patches are combined using WVMF. Results of this approach have been shown to overcome the FragTrack performance.

Even if fragment-based methods can be a good alternative to the general problem of object tracking, this kind of approach has not yet been applied to the problem of pedestrian tracking with calibrated cameras. In this paper, we argue that there are several different ways in which the calibration information can be used to improve tracking performance. We think that, in the context of surveillance systems, the use of calibrated cameras does not restrict too much our approach. Moreover, methods of self-calibration that work in these environments have been proposed [9] and can be combined with our framework.

III. Proposed Method

A. Patch creation

Splitting the target region into multiple patches can be used to solve ambiguities if a portion of the target is corrupted (e.g. because of a partial occlusion). If several patches are used, only some of them may be corrupted, and the motion of the uncorrupted patches can be used to robustly estimate the movement of the target.

One simple approach to create these patches is to split the region of interest into adjacent patches forming a rectangular grid, as proposed in [2] and [1]. In these methods, once the grid is created, it remains unchanged for the whole sequence (except for scale changes in [1]) and the computed displacement at each time step is the same for all patches. However, in general, a translational movement of an object in the world does not represent, in the image, the same translation for all patches. To address this issue, our method projects the 3D displacement vectors at different heights to compute the corresponding displacement for each patch. Therefore, a more loose arrangement of the patches is required. To this end, the proposed initialization procedure creates the patches in a way that their centers are vertically aligned in the WCS, what is expected in pedestrian tracking. The procedure is defined as follows.

First, the user is asked to indicate the point corresponding to the foot of the person of interest. Then, assuming that the camera parameters are known and that the ground plane is the xy-plane with $z = 0$, the subject position in the ground can be approximated with Equation (1).

$$w \begin{bmatrix} X_f \\ Y_f \\ 1 \end{bmatrix} = H^{-1} \begin{bmatrix} u_f \\ v_f \\ 1 \end{bmatrix}, \tag{1}$$

where H is the homography between the image plane and the scene ground plane, $[u_f, v_f]^\mathsf{T}$ corresponds to the point informed by the user and the point $[X_f, Y_f]^\mathsf{T}$ corresponds to an estimate of the subject translation in the world coordinate frame.

In the next step, the user is asked to inform the point in the top of head. The algorithm projects in the image a 3D-line with $x = X_f$, $y = Y_f$ and $z > 0$ using the camera projection matrix. Finally, the user selects the head position along this line and the patches are created such that their centers lay on the line that connects the head and foot of the subject. The total number of patches created at initialization is $\lfloor d_y/n_v \rfloor$, i.e. the ratio between the differences of the Y coordinates of the head and foot points, called d_y, and a minimum vertical size for the patch defined by the user, called n_v. In this work, we use $n_v = 20$, set empirically. The width of the patches is also defined by the user.

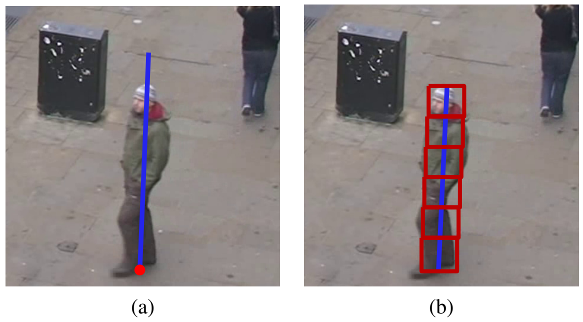
Fig. 1. Initialization procedure. (a) Selection of the foot point (red dot), and its vertical expansion (blue line). (b) Selection of the head point and patch creation (along a vertical line in the WCS).

Figure 1 shows the two main steps of the initialization procedure. This initialization makes our tracker more adaptable to different camera setups, because the algorithm does not assume that a person will appear in an upright position in the image, but instead it assumes that this orientation happens in the world coordinate frame. As it can be observed in Figure 1, the patches are not placed onto a regular grid, as opposed to [2], [1].

Finally, the Z coordinate of each patch center is computed using the ground position $[X_f, Y_f]^\mathsf{T}$ computed with Equation (1) and the camera projection matrix P. Latter, we refer to this value as the height H_i of a given patch i. Even though our tracker requires manual initialization, the procedure can be automatized. For instance, a person detector [8] can be used to define the subject bounding box and a background segmentation technique such as [10] can be employed to generate the mask in which the foot and head points can be easily extracted. However, the utilization of these techniques is beyond the scope of this work.

B. World WVMF

The patches created using the technique described in the previous section are tracked individually and a set of displacement vectors, one for each patch, is obtained. In this context, the purpose of the WVMF is to combine these vectors in a robust way to determine the displacement of the whole template. This work extends the method proposed in [2] by considering the displacement vectors in the world coordinate frame. Our main idea is that, if the height (in the WCS) of a patch is known, it is possible to reconstruct the 3D displacement vector of this patch in the WCS. Let $\boldsymbol{c}_i = [u_i, v_i]^\mathsf{T}$ be the center point of patch i in image coordinates and $\boldsymbol{d}_i = [\Delta u_i, \Delta v_i]^\mathsf{T}$ the associated displacement vector provided by a tracking procedure. In order to reconstruct $\boldsymbol{d}_i$ in the WCS, it is necessary to first reconstruct the point that represents the displaced patch center $\boldsymbol{m} = \boldsymbol{c}_i + \boldsymbol{d}_i$ in the world. This 3D point is latter referred as $\boldsymbol{M} = [X_i, Y_i, Z_i]^\mathsf{T}$. We assume that any displacement vector $\boldsymbol{d}_i$ corresponds to a translational displacement in the world that is parallel to the ground plane. Therefore, Z_i is set to a fixed value, namely the height H_i of the patch computed at initialization. The projection of $\boldsymbol{M}$ in the image plane is given by equation (2).

$$w \begin{bmatrix} u_i + \Delta u_i \\ v_i + \Delta v_i \\ 1 \end{bmatrix} = P \begin{bmatrix} X_i \\ Y_i \\ H_i \\ 1 \end{bmatrix}, \tag{2}$$

where P is the 3-by-4 projection matrix and w is the projection scale parameter. With the analysis of equation (2), it is possible to assert that X_i and Y_i are obtained by solving a simple linear system of two equations and two unknowns. Finally, the displacement vector $\boldsymbol{D}_i$ in the WCS associated with the patch i is given by the difference between the reconstructed point $[X_i, Y_i]^\mathsf{T}$ and the world point associated with the original patch center $[u_i, v_i]^\mathsf{T}$ and reconstructed using the same technique.

In perfect conditions, all the displacement vectors in the WCS will be very similar. However, as previously mentioned, partial occlusions, patches in uniform regions and illumination changes may corrupt the displacement vectors of some patches (and the corresponding reconstructed vectors in the world). Because of these outliers, computing the mean displacement vector would be a very naïve approach. A more adequate technique is the use of Weighted Vector Median Filters (WVMF), which implicitly account for outlier rejection. Let us denote, for simplicity reasons, $\boldsymbol{v}_i$ as the displacement vector $\boldsymbol{D}_t^i$ at time t. Given a set of N displacement vectors (where N is the number of patches), the first step of WVMF consists of computing the distance from each vector to all others:

$$s_j = s(\boldsymbol{v}_j) = \sum_{i=1}^{N} \|\boldsymbol{v}_j - \boldsymbol{v}_i\|, \quad j = 1, ..., N, \tag{3}$$

where $\|\cdot\|$ is a vector norm (in this work, we employed the L_2 norm). The filtered vector $\boldsymbol{v}_f$ is then defined according to

$$\boldsymbol{v}_f = \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} w_i \boldsymbol{v}_i, \tag{4}$$

where $w_i = f(s_i)$, and f is a nonnegative monotonically decreasing function (so that vectors that are farther from the median are associated with smaller weights).

The weights w_i in the original WVMF formulation [11] include only geometrical distances between pairs of vectors. We use a modification of the weights w_i, proposed in [2], that also includes the matching error b_i of each patch in the filtering process, so that patches with smaller matching errors carry more weight. More precisely, the proposed weights for the WVMF are given by

$$w_i = g(s_i, b_i), \tag{5}$$

where $g(x, y)$ is a nonnegative monotonically decreasing function when considering the variables x and y individually. With this choice for w_i, vectors that present smaller matching errors b_i and that are also geometrically consistent with the remaining vectors (i.e., present a smaller distance s_i) are prioritized in the weighted average. As proposed in [2] $g(x, y)$ is defined as an exponential function:

$$g(s_i, b_i) = e^{-\left[(s_i/\beta)^2 + (b_i/\gamma)^2\right]}, \tag{6}$$

If we increase the values of β and γ too much in equation (6), the filtered vector will be close to the mean of the displacement vectors. However, the decay of g should be strong for vectors with distance values far from the minimum. So, following the idea in [2], the β parameter is defined adaptively as the minimum value of the distances s_i at each time step. The γ parameter depends on the matching technique that is being used. A simple approach is to define γ as the maximum matching error value observed for non-occluded patches.

In Section III-C two different matching schemes are proposed, that are based on two different kinds of features: statistical features descriptors and color histograms. Even if the insertion of other features is beyond the scope of this paper, it is very easy to accommodate other methods in our framework, provided that it is possible define a matching distance between two image regions.

After applying the WVMF, the displaced points in the WCS are projected back onto image using the heights H_i of the patches; these projections correspond to the new patch centers in the image. This is done because, as mentioned before, even if the points have the same X and Y coordinate in the world, their projections does not necessarily constitute a line in the upright orientation. The effect of moving patches in this manner is that their centers will always correspond, in the world, to a 3D line that is perpendicular to the ground plane. Therefore, the tracker automatically adjusts the orientation among the patches at each time step in a way that is coherent to what is observed in the image.

Scale estimation: The estimation of the scale parameter is done through a very simple and yet efficient procedure that works independently of the WVMF algorithm. At initialization, the lower left and right corners of the lower patch (that are supposedly on the ground plane) are reconstruct in the WCS using equation (1). The distance between these corners is also stored. Then, at a given frame, the reconstructed points are first displaced using the filtered vector and then projected in the image. The scale at the current time step is obtained by computing the ratio between the corners distance computed at initialization and the distance of these projected points, since the distances in world coordinates should remain unchanged.

C. Patch Matching

There are several methods for computing the similarity between two image regions. In the WVMF framework different methods can be used, with the only requirement that a distance metric is provided to evaluate the similarity between two image regions. In this paper we propose two different kinds of image features to be used with the proposed method: statistical features and color histograms.

1) Statistical features: Statistical features descriptors have been used in recent years for tracking [12], [2] because this kind of descriptors require a small amount of memory and matching metrics that are cheaper to implement. Moreover, different types of features can be easily combined to increase performance.

For real-time tracking, the choice of features and statistical descriptors must reach an appropriate balance between discriminability and computational efficiency. We follow the work proposed in [2] in which RGB pixel values of a given region i are used to estimate a mean vector $\boldsymbol{\mu}_i$ and a covariance matrix C_i. Notice that this choice results in a compact representation of the region. Because the covariance matrix has only $d(d+1)/2$ distinct parameters, where d represents the dimension of the feature, only 9 parameters (6 for C_i and 3 for $\boldsymbol{\mu}_i$) are necessary to describe any image region.

The Bhattacharyya distance is a popular choice for comparing distributions based on the first and second order parameters. This distance for a region candidate i is given by

$$\begin{aligned} b_i &= \frac{1}{8}(\boldsymbol{\mu}_i - \boldsymbol{\mu}_m)^T \left[\frac{C_i + C_m}{2}\right]^{-1} (\boldsymbol{\mu}_i - \boldsymbol{\mu}_m) + \\ &+ \frac{1}{2}\ln\left(\frac{|(C_i + C_m)/2|}{\sqrt{|C_i||C_m|}}\right), \end{aligned} \tag{7}$$

where C_m and $\boldsymbol{\mu}_m$ are the model covariance matrix and mean vector, respectively. As proposed in [2], we used $\gamma = 0.15$ in Equation (6).

2) Color histograms: Histograms have been used in the past to model people appearance, such as in [13], [1]. Given a set of histograms computed at initialization, the objective is to find, at the each time step, the region with the most similar histograms. To compute the histogram, first the image is converted to the CIELab space, which separates the lightness of the color (L-channel) from the color channels (a and b); only the a- and b-channels are used because of their robustness to illumination changes.

For efficiency reasons, we assume that these two channels are independent, so a given image region is described by two histograms with N_b bins, normalized such that $\sum_{j=1}^{N_b} h(j) = 1$. To evaluate a given region candidate i, with associated histograms h_i^a and h_i^b (for the a- and b-channels, respectively) we use the Bhattacharyya distance between the candidate histograms and the model histograms [13]:

$$b_i = \frac{1}{2}\left(\sqrt{1 - BC(h_m^a, h_i^a)} + \sqrt{1 - BC(h_m^b, h_i^b)}\right), \tag{8}$$

where h_m^a and h_m^b are the model histograms and BC is the Bhattacharyya coefficient defined as follows:

$$BC(h_1, h_2) = \sum_{j=1}^{N_b} \sqrt{h_1(j) h_2(j)}. \tag{9}$$

In our implementation we set the number of bins to $N_b = 64$ and the model histograms are computed at initialization and remain unchanged for the entire sequence. The γ parameter of Equation (6) was set empirically to $\gamma = 0.25$. In the experiments we carried out, the use of color histograms resulted in a better tracking performance than statistical features (see Section IV). Moreover, it is possible to compute the histograms in a very efficient manner using the integral histogram data structure, proposed in [14].

D. Search Region

In order to individually track the patches at each time step, it is necessary to generate a set of candidate positions. One simple way of creating these candidates is to determine a fixed region around the previous position of the patch and then exhaustively search for the candidate that minimizes the matching distance [1], [2]. This approach has two important drawbacks.

On one hand, if the matching procedure is computationally expensive and a real-time performance is required, the size of the search region must be reduced. For this reason, the tracker will not be able to cope with large inter-frame displacements.

On the other hand, if the search region is too big for a given frame, a patch can be wrongly matched with false positives nearby, and it should be also mentioned that the incidence

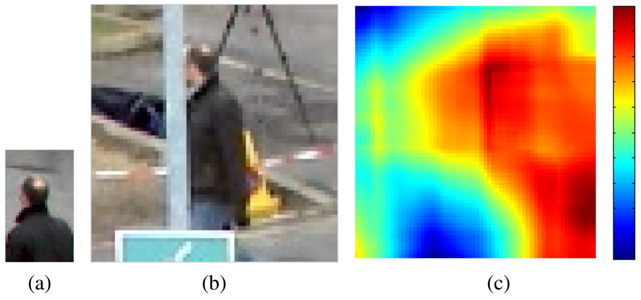
Fig. 2. (a) Example patch extracted at initialization. (b) The seacrh region. (c) Vote map using the technique described in III-C, illustrating false positives (lower right).

of false positives depends directly on the type of features that is being used. Figure 2 shows a vote map that was built using the method based on color histograms presented in Section III-C. It is possible to infer from the figure that, in some cases, it would be good to limit the search to avoid areas with false positives. Given that the maximum inter-frame displacement of a person in world coordinates is known, the corresponding displacement in image coordinates will depend on the distance between the person and the camera, suggesting that the position of the person must be taken into account to create the search region. We propose a simple procedure that adaptively creates the search region using the parameters of the camera and an estimate of the translation of the subject in the WCS.

The proposed method projects a region of the world in the image plane and samples this region to generate candidates. The size of the region can be defined using the frames per second fps value of the video and the maximum walking speed w that is expected to happen in the sequence. Then, the maximum inter-frame displacement in the WCS is $r = w/fps$, if both quantities are given in the same unit of length. To make the tracker more adaptable to different situations and to allow it to recover from failure, we introduce a relaxation parameter such that $\hat{r} = r + \alpha_r r$. In this work, we set the maximum speed parameter to $w = 1.5m/s$ and the relaxation parameter α_r was set to 0.5.

In order to simplify the sampling phase, the algorithm represents the region of possible translations as a square with side $s = 2\hat{r}$, which encloses a circle with the maximum possible displacement. Motion prediction was used to determine the center point, in world coordinates, of the search region. In this work, we used the Double Exponential Smoothing technique [15] which is very efficient and has a prediction performance equivalent to Kalman filters. A predicted displacement vector $\boldsymbol{v}_{t+\tau}$ is obtained with the following expression:

$$\boldsymbol{v}_{t+\tau} = \left(2 + \frac{\alpha\tau}{1-\alpha}\right)\boldsymbol{S}\boldsymbol{v}_t - \left(1 + \frac{\alpha\tau}{1-\alpha}\right)\boldsymbol{S}\boldsymbol{v}_t^{[2]}, \quad (10)$$

where α was set to 0.05 and $\tau = 1$; $\boldsymbol{S}\boldsymbol{v}_t$ and $\boldsymbol{S}\boldsymbol{v}_t^{[2]}$ are auxiliary variables computed through

$$\boldsymbol{S}\boldsymbol{v}_t = \alpha\boldsymbol{v}_t + (1-\alpha)\boldsymbol{S}\boldsymbol{v}_{t-1}, \quad (11)$$

$$\boldsymbol{S}\boldsymbol{v}_t^{[2]} = \alpha\boldsymbol{S}\boldsymbol{v}_t + (1-\alpha)\boldsymbol{S}\boldsymbol{v}_{t-1}^{[2]}. \quad (12)$$

Then, the center position of the search region square is the translation point of the person in the WCS plus the predicted vector $\boldsymbol{v}_{t+\tau}$. As previously mentioned, the translation point can be computed using expression (1), where u_f is the x coordinate of the center of the lower patch and v_f is the maximum value for the y coordinate of the same patch. Figure 3 shows the projected region in two distinct frames. As the number of candidates in a region is proportional to its size when projected, this value is not constant over time (the search region is larger when the target is closer to the camera, and smaller when it is far from the camera).

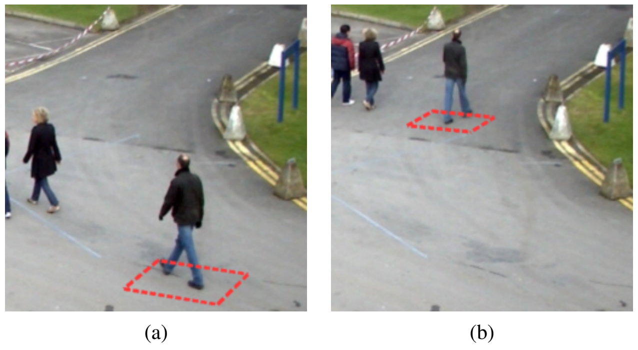
Fig. 3. Search regions for a subject in two different frames (red dashed line).

IV. Experiments

In this section we present several experimental results obtained with the proposed algorithm. Validation was performed qualitatively, by visual inspection of tracking results, and also quantitatively, by computing the error in tracking estimations using ground truth data. First, we evaluate the difference in performance between our tracker and the similar method proposed in [2]. Additionally, a comparison with two state-of-the-art techniques, namely the FragTrack algorithm [1] and the TLD tracker [16], is performed.

To perform the experiments we used two different datasets commonly employed to evaluate state-of-the-art methods: the popular *PETS* 2009 dataset [17] (sequence "S2.L1") and a sequence we latter refer as *TownCentre* [7]. Both datasets contain the calibration parameters of the camera and the annotated locations of the people in the scene[1].

In the *PETS* dataset, seven synchronized cameras are available for multi-view tracking so, as we are interested in the monocular case, a single view was used ("View001"). The sequence "S2.L1" contains challenging situations that are likely to occur in real scenarios such as occlusions, changes in scale, intersection of people with similar appearance, among others. The sequence has a low frame rate (fps = 7) which is a common feature of many surveillance cameras and can represent a difficulty to tracking algorithms since the inter-frame distances of the pedestrians tend to be large. The

[1]The ground truth for the PETS 2009 dataset was kindly provided by Anton Andriyenko at http://www.gris.informatik.tu-darmstadt.de/~aandriye/data.html

TownCentre sequence is a video from a busy town street that has an average of sixteen people visible at the same time. The frames in this dataset are high definition (1920×1080) and the frame rate is 25 fps.

To better analyze the behavior of the trackers, we selected different subjects in these datasets as our target. For the *PETS* dataset, we tracked three different subjects, illustrated in the first three rows of Figure 8. Each person represents a different level of difficulty for the tracking procedure:

- Subject 01: the person walks in a straight line and is partially occluded by a pole that is in the middle of the view. The subject has a very discriminant appearance.
- Subject 02: this subject walks almost in a straight line and crosses several people in his path. In the middle of the sequence, the subject crosses two people (one of them, with similar appearance) and, immediately after, appears occluded by the pole.
- Subject 03: this constitutes the more challenging scenario. The subject presents a very erratic path and large scale changes are present in the sequence. Once again, a critical point in the sequence is when the subject is occluded by the pole. However, in this case the subject is more distant from the camera.

In the following experiments, we used two distinct metrics to analyze the tracking errors. The first metric is defined as the Euclidean distance (in pixels) between the center point of the bounding box estimated by the tracker and the actual position. The second metric was used to evaluate the tracker estimates by computing the subject position on the ground plane in the WCS. The point at the bottom of the bounding box and the homography of the ground plane are used to extract the position using Equation (1). The error is then defined as the distance (in centimeters) between the ground position obtained with the tracking results and the position obtained with the ground truth.

The first set of experiments using the *PETS* sequence was intended to evaluate our contributions with respect to a similar method proposed in [2], called CPD tracker. For the CPD tracker we define a search region of 50×50, which is a larger region than the one used in the authors experiments [2]. This change was made because, with the suggested search region (30×30), the tracker tends to lose the target at the first few frames because of the sequence low frame rate combined with the fact that the target is relatively close to the camera, generating larger inter-frame distances. The set of features used in the CPD was composed of 3 RGB color channels. The update procedure proposed by the authors was removed after some preliminary experiments showed that the tracker estimation diverges from the true position when it is enabled. We compared the original CPD tracker to two versions of the proposed method: the first version uses statistical features (the same used in the CPD tracker) and the second uses color histograms as features (see Section III-C2). Figure 4 shows a comparison in the *PETS* sequences using the metric based on world translation that was previously described. The average errors (together with the standard deviation and the median value) for both metrics are shown in Table I.

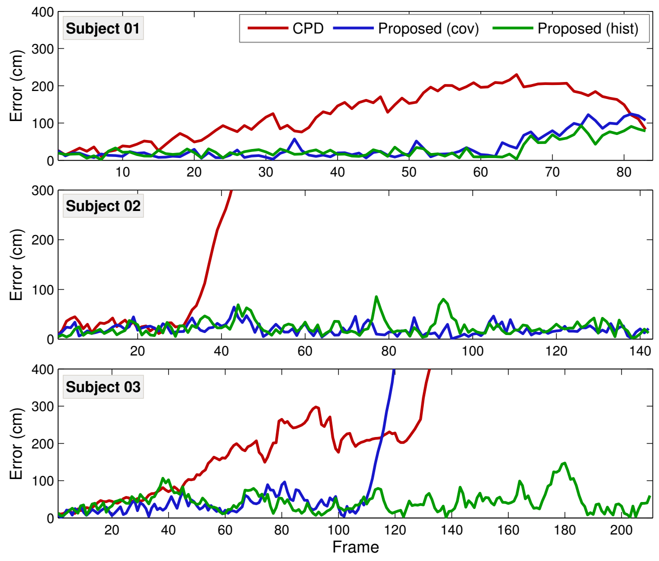

Fig. 4. Tracking errors (in mm) for the *PETS* sequence for the three target subjects. The error values for the CPD tracker [2] are depicted in red. Two versions of the proposed method are presented: errors of the version that uses statistical features are shown in blue while errors of the color histogram version are shown in green.

It is possible to observe that even when the same matching function was used to track the patches, our method presents better results. This difference is larger when we consider the translation errors, mainly due to two reasons. First, the CPD tracker does not estimate scale so the ground point is wrongly estimated when scale changes are present. Second, because our method combines the displacements in the WCS as opposed to the image coordinate system, the resulting displacement is more coherent w.r.t. the world. Furthermore, because our approach creates the patches candidates in a spatially-coherent way, much less candidates are needed. As an example, Figure 5 shows the number of candidates created at each frame for the sequence of Subject 3 of the *PETS* dataset. In this experiment, our tracker created an average of around 75 candidates for each patch. This constitutes 33 times less than the number of candidates used by the CPD tracker.

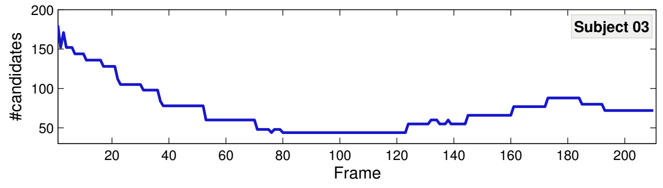

Fig. 5. Total number of candidates evaluated (per patch) at each frame for the sequence of Subject 03 of the *PETS* dataset. The value is related to the scale estimates computed at each frame and consequently it is not constant over time.

The proposed method also shows better results than other state-of-the-art methods. Figure 6 shows tracking results for our method (with color histograms matching), the FragTrack algorithm and the TLD tracker. We used the implementations provided by the authors in their websites and the parameters were set to the values suggested by them. An exception was the size of the window search used in the FragTrack that was set to 30 × 30. Notice that in some frames the error of TLD tracker is increased because the tracker detects another subject with similar appearance. The same configurations were used to perform another set of experiments in the *TownCentre* sequence. Two subjects were used as target and the results are shown in Figure 7. The performance of each method for all the sequences is summarized in Table I.

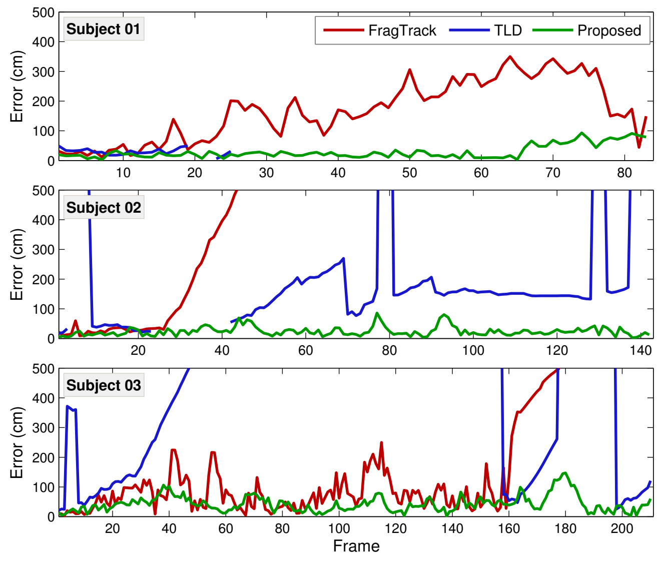

Fig. 6. Translation errors (in centimeters) for the *PETS* sequence for the three target subjects. Our method (in green) outperforms both the FragTrack algorithm [1] and the TLD tracker [16].

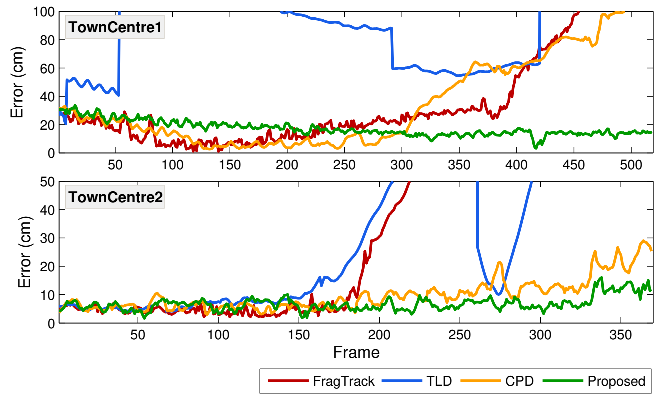

Fig. 7. Translation errors (in centimeters) for the *TownCentre* sequence.

Although continuously detecting the object can help the tracker to recover from failure, in the case of pedestrian tracking where different subjects can have similar appearances, a strong temporal consistency is required. On the other hand, even if fragment-based methods can cope with partial occlusions, two important things are necessary to accurately estimate the position during and after the occlusion: the position just before the beginning of the occlusion (e.g. around frame #160 for subject 03) must be accurate and the tracker must avoid the false positives nearby that have greater influence when an occlusion is happening. The FragTrack and the CPD tracker fail to accomplish both. First, neither of methods estimate scale[2] so the position before an occlusion can already suffer from error accumulation. Second, since the sizes of the search region in these methods are constant, the estimate tends to be drawn to a nearby false positive.

For the two experiments involving the *TownCentre* sequence, our method shows average errors of around 17cm and 6cm, respectively. For illustration, the results of the method recently proposed by Benfold and Reid [7] generated an error of around 3cm for both sequences. Although their method outperforms ours, it must be noticed that our approach constitutes a causal method, i.e. only past and current information are used to compute the estimates. On the contrary, their method also includes future information in the estimation phase which introduces a significant latency to the system (equal to four seconds, according to [7]).

Figure 8 shows two illustrative frames for each of the five experiments showed in our paper, along with the resulting bounding boxes of the evaluated methods. Our approach is able to correctly estimate the subject position and scale in these sequences even after severe occlusion. It is also possible to see that our simple and direct procedure for scale estimation provide good results, despite the large scale changes that occur in these datasets. The complete videos of the five sequences can be found in the supplementary material.

In order to examine the impact of a bad initialization, we tested multiples times the tracker adding Gaussian noise in the foot and head locations at initialization. In most cases, the proposed approach was able to correctly track the target despite the error introduced by the different initialization points. However, when the initial patches contain a large part of the background, the tracker sometimes fails in the frames where the subjects are partially occluded. To mitigate this problem, a background segmentation technique could be applied to identify the portions of the initial template that belong to the background.

Even if the proposed method works well in most scenarios, if fails in the presence of total occlusions that last for several frames. To overcome this limitation, a reinitialization of the target location based on a people detector could be employed. Another current limitation of the proposed method is that the initial template is not updated during the sequence, so if the appearance change drastically overtime, the method might lose track of the target.

V. Conclusion

In this work we presented a robust patch-based approach to pedestrian tracking. Our method combines the estimates of individually tracked patches using a Weighted Median Filter Vector computed in the world coordinate frame. We also

[2]The FragTrack implementation provided by the authors at http://www.cs.technion.ac.il/~amita/fragtrack/fragtrack.htm does not include scale estimation. However, a scheme to compute scale is indeed proposed in [1].

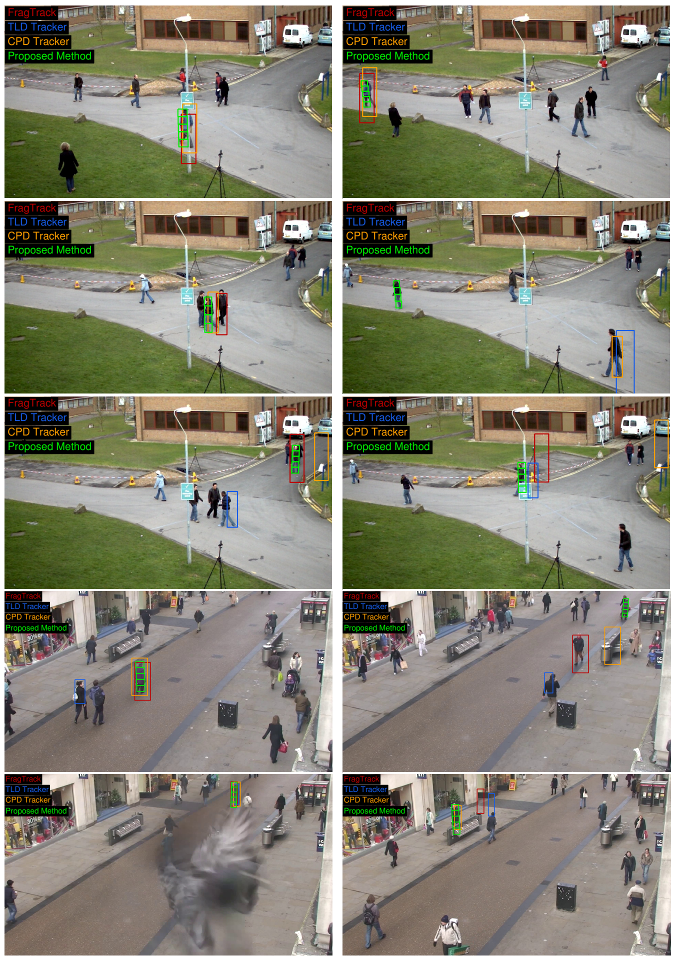

Fig. 8. Example frames and results of the five sequences used in experiments.

propose the utilization of the calibrated camera parameters to directly compute the scale and to limit the search region, as well as to compute the translational displacement vector in world coordinates. The resulting system is able to handle occlusions and scale changes while reducing the computational cost w.r.t. state-of-the-art techniques.

There are several interesting avenues for current and further work. First, we intend to investigate the use of different features in our framework that can improve tracking performance. We are also interested in the implementation of a fully automatic initialization of our tracker, similar to the detection method proposed in [8]. However, in our framework, we need to combine detection with background segmentation to extract the subject head and foot points. Additionally, it would be interesting to couple our approach to a self-calibration method such as the one proposed in [9] to expand our algorithm to be used with uncalibrated cameras.

Acknowledgment

The authors would like to thank Brazilian agencies CNPq and Capes for supporting this work.

TABLE I
TRACKING RESULTS USING THE TWO METRICS DISCUSSED.

Sequence	Metric		FT	TLD	CPD	Our
PETS S01	2D (px)	Average Error	12.08	13.48	9.83	**4.78**
		Std. Deviation	6.01	7.0	4.25	**3.24**
		Median	10.95	12.02	10.22	**3.95**
	3D (cm)	Average Error	168.25	29.55	123.75	**28.82**
		Std. Deviation	100.17	**10.79**	64.78	22.68
		Median	168.5	29.61	128.74	**20.67**
PETS S02	2D (px)	Average Error	189.73	46.97	277.02	**5.36**
		Std. Deviation	197.57	106.66	187.96	**3.0**
		Median	120.65	13.59	296.54	**4.53**
	3D (cm)	Average Error	419.52	224.55	784.8	**23.70**
		Std. Deviation	431.32	299.66	536.85	**15.35**
		Median	253.04	151.6	883.4	**20.07**
PETS S03	2D (px)	Average Error	57.64	100.85	155.04	**5.41**
		Std. Deviation	77.78	79.75	189.78	**3.64**
		Median	22.11	95.29	35.34	**4.35**
	3D (cm)	Average Error	189.13	632.08	529.82	**42.47**
		Std. Deviation	222.9	436.55	532.26	**27.36**
		Median	83.05	790.64	228.04	**37.84**
TownCentre1	2D (px)	Average Error	89.29	329.04	80.27	**22.33**
		Std. Deviation	133.33	189.73	76.37	**8.4**
		Median	30.15	309.67	42.99	**20.86**
	3D (cm)	Average Error	39.58	109.34	35.43	**17.26**
		Std. Deviation	43.84	59.11	31.77	**5.18**
		Median	22.88	97.03	22.65	**15.56**
TownCentre2	2D (px)	Average Error	208.35	213.37	22.21	**17.09**
		Std. Deviation	256.74	316.12	7.79	**6.39**
		Median	28.97	63.93	20.89	**16.08**
	3D (cm)	Average Error	66.25	67.57	9.72	**6.75**
		Std. Deviation	81.17	102.69	**5.56**	24.9
		Median	11.47	16.80	8.0	**6.37**

References

[1] A. Adam, E. Rivlin, and I. Shimshoni, "Robust fragments-based tracking using the integral histogram," in *IEEE CVPR*, vol. 1, 2006, pp. 798–805.

[2] L. L. Dihl, C. R. Jung, and J. C. Bins, "Robust adaptive patch-based object tracking using weighted vector median filters," in *SIBGRAPI*, 2011, pp. 149–156.

[3] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM CSUR*, vol. 38, no. 4, pp. 1–45, 2006.

[4] I. Haritaoglu, D. Harwood, and L. Davis, "W4 s: A real-time system for detecting and tracking people in 2 1/2d," *ECCV*, vol. 1406, pp. 877–892, 1998.

[5] D. Ramanan, D. Forsyth, and A. Zisserman, "Tracking people by learning their appearance," *IEEE PAMI*, vol. 29, no. 1, pp. 65–81, 2007.

[6] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, "Multi-camera people tracking with a probabilistic occupancy map," *IEEE PAMI*, vol. 30, no. 2, pp. 267–282, 2008.

[7] B. Benfold and I. Reid, "Stable multi-target tracking in real-time surveillance video," in *IEEE CVPR*, 2011, pp. 3457–3464.

[8] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE CVPR*, vol. 1. Ieee, 2005, pp. 886–893.

[9] F. Lv, T. Zhao, and R. Nevatia, "Self-calibration of a camera from video of a walking human," in *IEEE CVPR*, vol. 1, 2002, pp. 562 – 567 vol.1.

[10] S. Kwak, T. Lim, W. Nam, B. Han, and J. Han, "Generalized background subtraction based on hybrid inference by belief propagation and bayesian filtering," in *IEEE ICCV*, 2011, pp. 2174–2181.

[11] J. Astola, P. Haavisto, and Y. Neuvo, "Vector median filters," *Proceedings of the IEEE*, vol. 78, no. 4, pp. 678–689, 1990.

[12] F. Porikli, O. Tuzel, and P. Meer, "Covariance tracking using model update based on lie algebra," in *IEEE CVPR*, vol. 1, 2006, pp. 728–735.

[13] J. Gall, B. Rosenhahn, T. Brox, and H. Seidel, "Optimization and filtering for human motion capture - a multi-layer framework," *IJCV*, vol. 87, no. 1, pp. 75–92, 2010.

[14] F. Porikli, "Integral histogram: A fast way to extract histograms in cartesian spaces," in *IEEE CVPR*, vol. 1, 2005, pp. 829–836.

[15] J. LaViola, "Double exponential smoothing: an alternative to kalman filter-based predictive tracking," in *EGVE*, 2003, pp. 199–206.

[16] Z. Kalal, J. Matas, and K. Mikolajczyk, "Pn learning: Bootstrapping binary classifiers by structural constraints," in *IEEE CVPR*, 2010, pp. 49–56.

[17] J. Ferryman and A. Ellis, "Pets2010: Dataset and challenge," in *IEEE AVSS*, 2010, pp. 143–150.