Hierarchical Grid-based People Tracking with Multi-camera Setup

Lili Chen, Giorgio Panin, and Alois Knoll

Department of Informatics, Technische Universität München,
85748 Garching bei München, Germany
{chenlil,panin,knoll}@in.tum.de

Abstract. We present a hierarchical grid-based tracking methodology for multiple people tracking in a multi-camera setup. In this system, frame-by-frame detection is performed by means of hierarchical likelihood grids, by matching shape templates through an oriented distance transform over foreground intensity edges, followed by clustering in pose-space. Subsequently, multi-target tracking is achieved by means of global nearest neighbor data association, with a fully automatic initialization, maintainance and termination strategy. We demonstrate our system through experiments in indoor sequences, using a four-camera calibrated setup. Moreover, in the present paper we present the improvements obtained by means of a fast algorithm for computing the oriented DT, as well as using multi-part shape templates in place of a simple cylinder model, for a more precise localization.

Keywords: edge-based background subtraction, hierarchical likelihood grids, oriented distance transform, multi-view and multi-target tracking

1 Introduction

Nowadays automatic visual surveillance is becoming increasingly popular, because of its wide applications in indoor and outdoor environments with security requirements. Usually there are two major problems in this system: one is to detect moving targets, and the other is to keep them tracked throughout the sequence. As the most representative application, detecting and tracking people is obviously the most challenging and attractive topic, due to people's huge variations in physical appearance, pose, movement and interaction. Therefore, people detection and tracking receives a significant amount of attention in the area of research and development.

Although some systems have been successfully developed towards this challenging task, it still remains difficult to detect and track multiple people precisely and automatically in a cluttered scene. This paper addresses the problem of employing a grid-based tracking-by-detection methodology. The primary goal is to develop a fully automatic system for tracking multiple people in an overlapping, multi-camera environment, providing a 3D output robust to mutual occlusion between interacting people.

As a commonly used technique for segmenting out objects of interest, background subtraction has achieved a significant success in fixed camera scenarios. Most of methods work by comparing color or intensities of pixels in the incoming video frame to the reference image [6, 7]. However, it has the drawback of being susceptible to illumination changes, and provides a less precise localization. In contrast, we propose here an edge-based background subtraction, which employs Canny edge map together with Sobel gradients, because edges are more precisely and stably localized, to a better extent in presence of illumination changes.

A second contribution of our system is frame-by-frame detection by means of hierarchical likelihood grids. This scheme, adapted from [25], takes the advantage of multi-resolution grids that can, precisely and efficiently locate targets in cluttered scenes, without prior knowledge of their position. In particular, we compute the likelihood by edge matching through a fast oriented distance transform, which extends from our previous work [26], speeds up the computation by performing multiple searches along the given orientation while matching not only the location of edge points but also their orientation. And the likelihood is first computed on a coarse grid, then refined on the next level only the locations where likelihoods are higher than a given threshold. Subsequently, we perform state-space clustering on the high-resolution grid, in order to find the relevant peaks, possibly associated to people.

The third main issue consists in associating detected peaks to tracks, which is a classic data association problem. Several approaches have been developed for this purpose, the most representative ones being [13, 14]; however, in place of complex methods, which require more complex models and parameter tuning, and further increase the computational complexity, our tracking module employs a Global Nearest Neighbor (GNN) approach in order to initiate, maintain and terminate tracks automatically.

The remainder of the paper is organized as follows: Section 2 reviews the state of the art and related work to our paper. Section 3 describes the general system overview with hardware setup and algorithmic flow of software. In Section 4, we provide the detailed detection procedure, including models, edge-based background subtraction, hierarchical grid evaluation as well as model-based contour matching and state-space clustering. Tracking by data association is presented in Section 5. The experimental results are discussed in Section 6. Finally, Section 7 summarizes the paper and proposes future development roads.

2 Related Work

A vast amount of literature has been published on people detection and tracking. We can mainly classify them into four categories: region-based approaches [21], based on the variation of image regions in motion; feature-based [17, 22], that usually utilize information about color, texture, etc.; contour-based [18, 19], that make use of the bounding contours to represent the target outline; and model-based methods [20, 23] that explicitly require a 2D or 3D model of a person. However, a too detailed review is beyond the scope of our paper, therefore, in

the following we will focus on people tracking-by-detection methodologies, more related to our work. There has been a number of literature on this approach [5, 8, 9], where detection of people in individual frame, as well as data association between detections, are the most challenging and ambiguous issues [4].

Template-based methods have yielded nice results for locating targets with no prior knowledge in a cluttered scene. In [1], the efficiency of this method is illustrated by using about 4,500 templates to match pedestrians in images. The core idea is using a Chamfer distance measure, so that matching a template with the DT image results in a similarity measure. Meanwhile this approach enables the use of an efficient search algorithm. However, if only computing the location of edge pixels without considering their orientation when computing distance transform, it inevitably leads to a high rate of false alarms in presence of clutter.

Another highlight of this system is the utilization of a template hierarchy, which is generated automatically from available examples, and formed by a bottom-up approach, using a partitioned clustering algorithm. It only searches locations where the distance measure is under a given threshold, so a speed-up of three orders of magnitude is demonstrated, compared to exhaustive searching.

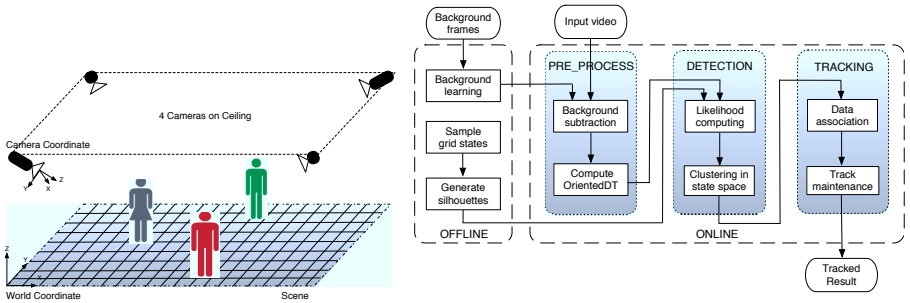
This idea was taken further by [25], that however does not build the template hierarchy (or tree) by bottom-up clustering, rather by partitioning a state-space represented with an integral grid. The grid is hierarchically partitioned as the search descends into each region, so that regions at the leaf-level define the finest partition. This method is demonstrated to be capable of covering 3D motion, even with self-occlusion. Unfortunately, both approaches need a very specific model, only valid for a specific target.

Once the measurements have been obtained from the frame-by-frame detection, data association can be applied to solve the problem of measurement to track assignment. A simple nearest-neighbor approach [12] uses only the closest observation to any predicted state to perform the measurement update, it is commonly used for MTT systems because of its fast computation. More complex approaches, such as Joint Probabilistic Data Association Filter (JPDA) [13] combines all of the potential measurements into one weighted average, before associating it to the track, in a single update. By contrast, Multiple Hypothesis Tracking (MHT) [14] calculates every possible update hypothesis, with a track, formed by previous hypotheses associated to the target. Both methods are known to be quite complex, and require a careful implementation in terms of parameters; in particular, the latter cannot avoid the drawback of an exponentially growing computational complexity, with the number of targets and measurements involved in the resolution situation.

3 System Overview

In this section, we describe the hardware setup and present an overview of our tracking system. The overall setup is depicted in Fig. 1(a). Four uEye usb cameras are mounted overhead on the corners of the ceiling, each of them observing the same 3D scene synchronously. Furthermore, all the four cameras are con-

nected to one multi-core PC. A necessary step before being able to get accurate 3D information, is calibration of the intrinsic and extrinsic camera parameters, that we perform with the Matlab Calibration Toolbox [1], with respect to a *world* coordinates system placed on the floor.



(a) (b)

Fig. 1. System overview. (a) Hardware setup. (b) Block diagram of the tracking system.

The tracking system is designed and implemented in the *OpenTL* framework[2] [2, 10], which is a structured, general purpose architecture for model-based visual tracking. The block diagram is provided in Fig. 1(b), that consists of two main processing modules. Offline, we use a certain number of background frames to learn the background model. Moreover, grid states are sampled for each level, and the silhouettes are generated by projecting the external contours of the cylinder shape and keeping, for each contour and each camera view, a list of pixel positions and normals. Online, we have three main sub-modules: pre-process, detection and tracking.

In the pre-process part, for each camera view foreground contours are segmented by edge-based background subtraction, using the learned model. Afterwards, we compute an oriented distance transform onto this image, in order to match, for each template, both the location and the orientation of its contours. In particular, the oriented DT is efficiently computed over a finite set of orientations, so that the image is sampled over parallel scan lines that are pre-computed. The advantage of using both edge position and orientation, during background subtraction as well as template matching, is a strong reduction of false alarms.

Detection part first computes the likelihoods by matching projected templates and oriented DT for each camera view, where the likelihoods are computed on the coarse grid firstly, then refined on the next resolution only the locations where the likelihood is higher than a given threshold, the joint likelihoods can simply be multiplied then. The object-level measurements, or target hypotheses, are obtained by means of likelihood grid clustering, performed by

[1] `http://www.vision.caltech.edu/bouguetj/calib_doc/`
[2] `http://www.opentl.org`

Gaussian filtering of the high-resolution grid, and local maxima detection. Finally, the tracking module performs measurement-to-target association with the Global Nearest Neighbor approach.

4 People Detection

In this section, we provide more details about people detection, that serves as one of our key building blocks for our system.

4.1 Construction of Template Hierarchy

The idea to construct a template hierarchy is inspired by the paper [25], as well as by the system developed by [1], extended to multiple views, multiple targets, and with a more general template.

Assuming there are L levels of search, the state space is partitioned with a coarse-to-fine strategy. A graphical illustration is shown in Fig. 2.

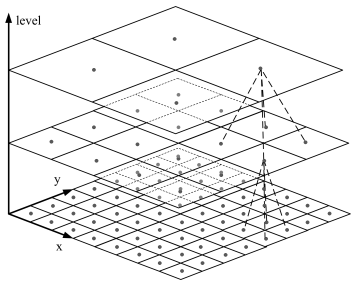


Fig. 2. Grid based state space with hierarchical partition.

Each discrete region $\left\{R^{i,l}\right\}_{i=1}^{N_l}$, where N_l is the number of cells at level l, is sampled at its center, before the template hierarchy is generated. Meanwhile, we connect regions at a child level with its parent cell, by computing the nearest-neighbor in state-space, as well as its nearest neighbors within the same level, as it will be described in Section 4.4, in order to smooth the grid likelihoods.

After sampling the grid, templates are generated by rendering the 3D model at each state, under the respective camera projection. To more precisely match our target, the model chosen here is composed of 3 cylinders, where one cylinder is for the head, one for the torso, and one for the legs. The model undergoes (x,y) translation on the floor, while its silhouette is generated by projecting the external contour. An example of the model and a partial view of the hierarchy of silhouettes are shown in Fig. 3.

For each silhouette, the position of each point as well as its normal is collected, as it will be described further in Section 4.3. As already emphasized, both grid sampling and template hierarchy generation are performed offline.

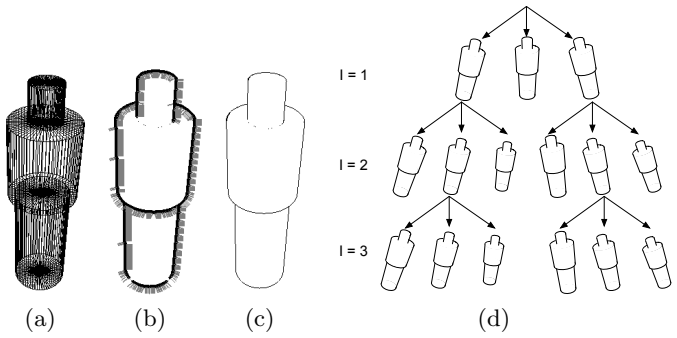


Fig. 3. Our model. (a) Discretized cylinder. (b) Silhouette with normals. (c) Silhouette without normals. (d) Hierarchy of the silhouettes.

4.2 Background Learning and Foreground Segmentation

In order to match the image data with the templates, we first apply an edge-based background subtraction.

This approach can be divided into two phases: background learning (offline) and foreground segmentation (online). In the first phase, we utilize a certain number N of frames without people, to learn the background model. Let $\Theta_b(t), G_{bx}(t), G_{by}(t)$ respectively be the Canny edge map, Sobel x-gradient and y-gradient images, detected at frame $I_b(t)$. The Canny map Θ_b is accumulated by binary OR, from frame $\Theta_b^{(I)}(1), \ldots, \Theta_b^{(I)}(N)$, while Sobel gradients are accumulated in a running average over the same frames. At the end, we normalize the accumulated Sobel image

$$G_{bx}^2 + G_{by}^2 = 1, \ \forall (x, y) \ . \tag{1}$$

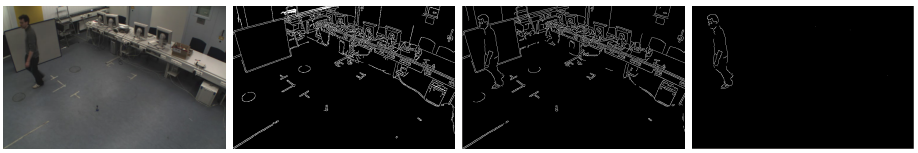
Subsequently, standard distance transform is applied to the accumulated background Canny map, and thresholded to a few pixels, providing a binary mask $\Theta_{DT} \in \{0, 1\}$, where potential background edges are found.

Online, from foreground Canny map and Sobel gradients $\Theta_f(t), G_{fx}(t), G_{fy}(t)$ of frame $I_f(t)$, we test the position and orientation of each edge pixel: edges $\Theta_f(t) \neq 0$ that lie near to a background edge $\Theta_{DT} \neq 0$ are candidate for removal. Then, we further test these edges for orientation with the Sobel masks, if the scalar product is higher than another threshold θ

$$\frac{G_{bx}G_{fx} + G_{by}G_{fy}}{\sqrt{G_{fx}^2 + G_{fy}^2}} > \theta \ . \tag{2}$$

the point is removed from $\Theta_f(t)$.

Fig. 4 shows an example of this procedure: as we can see, the resulting edge map robustly preserves the person contours, while discarding most of the background edges.



(a) (b) (c) (d)

Fig. 4. Edge-based background subtraction. (a) Original frame. (b) Learned background model. (c) Unsegmented foreground edge. (d) Segmented foreground edge.

4.3 Template Matching with Fast, Oriented Distance Transform

The next step is to match foreground edges with the model silhouettes. One possibility would be to use the Chamfer distance transform on the edge map, that is tolerant to small shape variations, and has already been applied in several works, such as [1, 24]. However, in case of images with considerable clutter, a significant rate of false alarms would be present. This problem can be reduced by matching not only the location of edge points, but also their orientation [3].

Therefore, an *oriented* distance transform, considering orientation of edge points, is more than necessary. It was first proposed in our previous work [26], where the oriented DT is defined by scanning the edge image through raster lines from top to bottom and from left to right. This method needs two scans for each raster line: one for finding edge pixels on the line, and the other for writing the DT values in the output image. In particular, all of the image pixels on each line must be read, before deciding whether any edge pixel is present, and then assign them DT values. However, if the line crosses no edge, no one of these pixels will have a valid DT value. Moreover, even for a valid scan line, most pixels have a DT which is higher than the validation gate, and therefore have no valid DT as well, but the line iterator can only proceed one pixel at a time, therefore wasting computational resources.

Here, we propose a significant faster implementation, that instead performs line scans starting directly from the *edge pixels*, and proceeding in both directions, until the desired distance. This is obtained by maintaining a double-linked, circular list of *exploring units*, two for each foreground edge pixel(which are in a limited amount, after background subtraction), that keep trace of the current DT value, and perform a single line iteration in each direction, execute one after the other through the circular list.

A single iteration consists of: one read operation, to check the current pixel, one write operation, and one increment of pixel position and DT value, for the next round. If a pixel has been already visited(i.e. its DT value is not infinity), or the DT value is beyond the validation gate, the unit is stopped and removed from the list, in order to not be checked again. By performing a single iteration per unit in a round, we make sure that two units, coming from two different edges but traveling along the same scan line in opposite directions, will meet exactly in the mid-point, and the DT values will be correctly assigned to the closest edge on the line.

When the list is empty, the algorithm terminates. Overall, this reduces the number of read/write/iterate operations to almost the minimum (only valid pixels are visited) except for the mid-point above mentioned, which will be *read* twice, by the two units that will terminate after each other. However, and in particular when the validation gate is reasonably small, this case is limited to a very small set of pixels. A pseudo-code of the fast oriented DT is shown in Algorithm 1, while Fig. 5 shows an example of results.

Algorithm 1 Fast oriented distance transform

Initialization :
Fill the DT image with ∞, *apart from* 0 *at the foreground edges.*
Create a double−linked, circular list of "exploration units", two for each foreground edge pixel, going in opposite directions (= *line iterators*).
Each unit consist of :
- *A distance counter*(*initialized with* 0);
- *A line iterator*(*initialized with edge pixel position*), *with a given direction*;

Main loop :
while *list is not empty* **do**
  Take current element of the list;
  Read DT value at (x, y);
  if $0 < DT(x, y) < \infty$ **then**
    Remove unit from the list;
  else
    Write the counter value into the DT image;
    Increment counter;
    if *counter* > *validation gate* **then**
      Remove unit from the list;
    else
      Increment line iterator;
    end if
  end if
  Move to the next unit in the list;
end while

Once DTs are computed, template matching simply amounts to compute the likelihood, by summing up all values over the silhouette pixels, in the corresponding direction of the normal. To formalize the idea, a projected template s is represented by a set of pixel positions and normals $\{x_i, y_i, g_i\}_{i=1}^{N}$, obtained by re-projection through a 3×4 camera projection matrix P, where g_i selects the nearest $\gamma \in \Gamma$, from which the DT value will be taken. Therefore, the likelihood for state hypothesis s is given by:

$$P(z|s) = \exp\left(-\frac{1}{2NR^2}\sum_{i=1}^{N}\min\left(DT_{\gamma(g_i)}(x_i, y_i)^2, D_{max}^2\right)\right) . \tag{3}$$

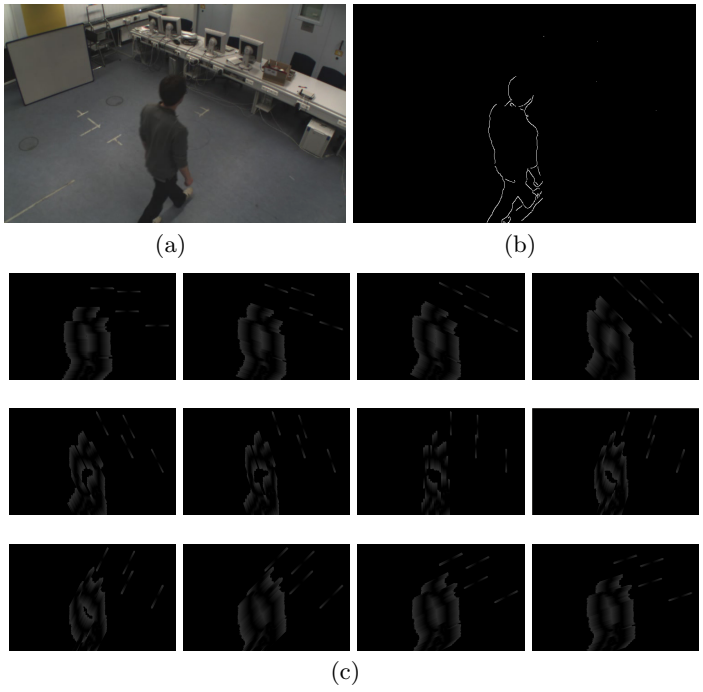


Fig. 5. Results of oriented distance transform. (a) Original image. (b) Foreground edge map. (c) Oriented DT results (at 12 discrete orientations).

where $\gamma(g_i)$ denotes the closest available direction to the normal, and the sum is performed over all values $\{x_i, y_i, g_i\}_{i=1}^{N}$. R is the measurement standard deviation, and an outlier threshold is usually fixed at $D_{max} = 3R$, which is our validation gate for a more robust matching. Also notice that, in order to avoid problems with different scales, the sum is further normalized by N.

During the computation of likelihood, a coarse-to-fine search strategy is applied by evaluating it, at each level, only for locations where the parent cell likelihood is higher than a given threshold, which is usually obtained as the average likelihood [25]. For those cells where the parent likelihood is under the threshold, its value is simply inherited, thus saving a large amount of computation.

4.4 Likelihood Grid Clustering

In order to obtain the object-level measurements, or target hypotheses, after likelihood computation we employ a clustering procedure on the high-resolution grid, where each cluster is a local maximum, potentially corresponding to a person.

This approach is similar to mean-shift, but explicitly done on discrete states. First of all, a Gaussian filtering is applied to the grid, where the isotropic Gaussian corresponds to the filtering kernel. For each cell s_i within the grid, we

take the nearest neighbor s_j by looking at the connected states with distance $d_{i,j} = \|s_i - s_j\|$ up to a validation gate $D_{max} = 3\sigma_s^2$, where σ_s^2 is the measurement covariance in *state-space*, these neighbors are pre-computed in the off-line phase. For each neighbor, the Gaussian weight is also pre-computed by

$$W_{i,j} = exp(-\frac{d_{i,j}^2}{2\sigma_s^2}) \; . \tag{4}$$

the computed weights are also normalized to 1, so that the smoothed likelihood for state cell s_i is given by

$$P(z|s)_{weighted}(i) = \sum_{i,j} W_{i,j} \cdot P(z|s)(j) \; . \tag{5}$$

Subsequently, local maxima are detected (within the same neighborhood), to obtain the target hypotheses, or measurements. The final step will be to associate these hypotheses to tracks, as it will be described in next section.

5 Multiple People Tracking

In this section we deal with the problem of multi-target tracking, by associating measurements obtained from our detector to individual tracks, also performing automatic track initiation and termination.

In particular, our track management follows a strategy indicated in [11]:

- *Track initiation* In case of new targets entering into the scene, they will generate measurements that are too far from the existing targets, and therefore can be used to start new tracks. In this case, they are labeled with a unique ID, and a counter for the number of consecutive, successful detections for this target is also initialized to 1.
- *Track maintainance* During tracking, a target is successfully detected whenever the data association algorithm provides one valid measurement for it, so its counter is increased up to a maximum value (which can be taken as a confirmation time), while in case of misdetection it will be decreased. Those targets which are successfully detected over the confirmation time, can be considered as stable targets and maintained by the algorithm. In this way, if a target is misdetected for a few frames in case of occlusion, it can still be recovered until the counter goes to 0.
- *Track termination* When a target exits the scene, or after occlusion for a too long time, its misdetection counter goes to 0, and its track is terminated.

A pseudo-code of the whole procedure is shown in Algorithm 2, where the GNN algorithm is called in (line 25).

The data association problem consists in deciding which measurement should correspond to which track. Although our detection algorithm is fairly robust, it is also not person-specific, and therefore in a small indoor environment there are always ambiguities, arising from neighboring targets, as well as from missing

Algorithm 2 Track management with GNN

```
if nMeasurements = 0 then
  for i = 0 to nTargets do
    DecreaseCounter(target[i]);
    if Counter(target[i]) > 0 then
      MaintainTarget(target[i]);
    else
      TerminateTarget(target[i]);
    end if
  end for
else
  if nTargets = 0 then
    for j = 0 to nMeasurements do
      newTarget = CreateTarget(meas[j]);
      ResetCounter(newTarget);
    end for
  else
    for i = 0 to nTargets do
      for j = 0 to nMeasurements do
        D(i, j) = Distance(target[i], meas[j]);
        if D(i, j) > ValidGate then
          D(i, j) = ∞;
        end if
      end for
    end for
    (i ↔ j) = GNN(D);
    for i = 0 to nAssocTargets do
      if D(i, j(i)) ≤ ValidGate then
        MoveTarget(target[i], meas[j]);
        IncreaseCounter(target[i]);
        if Counter(target[i]) > MaxC then
          Counter(target[i]) = MaxC;
        end if
      else
        DecreaseCounter(target[i]);
        if Counter(target[i]) = 0 then
          TerminateTarget(target[i]);
        end if
      end if
    end for
    for j = 0 to nUnassocMeas do
      newTarget = CreateTarget(meas[j]);
      ResetCounter(newTarget);
    end for
  end if
end if
```

detections and false alarms caused by background clutter. To this respect we employ the Global Nearest Neighbor (GNN) approach, that gives a good solution for this problem [15], while requiring relative low computational cost.

The first step of the GNN is to set up a distance (or cost) matrix: assuming that, at time t, there are M existing tracks and N measurements, the cost matrix is given by

$$D = \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1N} \\ d_{21} & d_{22} & \cdots & d_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ d_{M1} & d_{M2} & \cdots & d_{MN} \end{pmatrix} . \tag{6}$$

where d_{ij} is the Euclidean distance between track i and measurement j, and $i = 1, 2, \ldots, M; j = 1, 2, \ldots, N$. In particular, d_{ij} is set to ∞ if it exceeds the validation gate, which is a circle with fixed radius around the predicted position, eliminating unlikely observation-to-track pairs. Moreover, it is commonly required that a target can be associated with at most one measurement (none, in case of misdetection), and a measurement can be associated to at most one target (none, in case of false alarms).

The GNN solution to this problem is the one that maximizes the number of valid assignments, while minimizing the sum of distances of the assigned pairs. To this aim, we adopt the extended Munkres' algorithm [16], where the input is the cost matrix D, and output are the indices (row, col) of assigned track-measurement pairs.

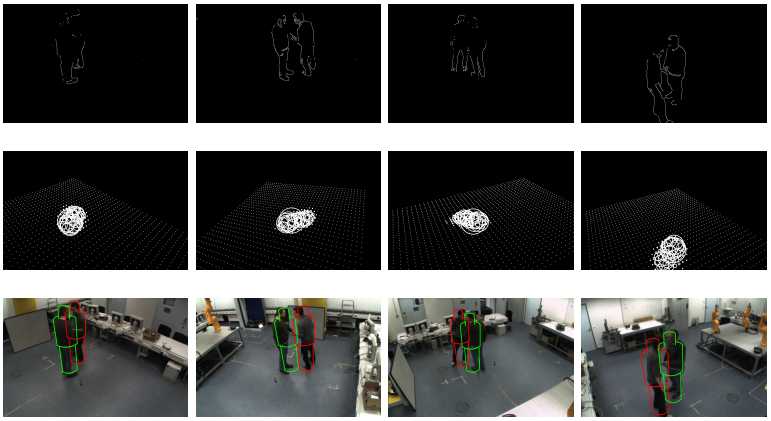
6 Experimental Results

We evaluated the proposed algorithms through pre-recorded video sequences, with multiple people entering and leaving the scene, as well as interacting with each other. The sequences have been simultaneously recorded from four cameras, as described in Section 3, with a resolution of (752×480), and a frame rate of 25 fps.

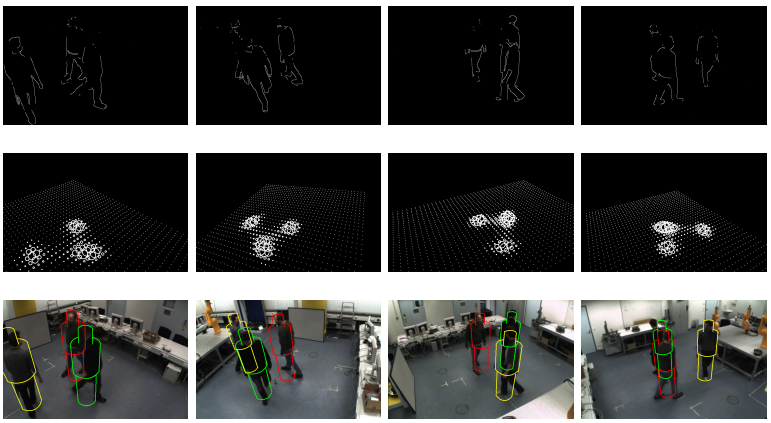
Before carrying out detection and tracking, state grids are set up at all levels, respectively 10×10, 20×20 and 40×40 from the coarsest to the finest, resulting in a total of 2100 grid cells, and the same amount of silhouette templates are sampled off-line. Since the area of interest is $(6m \times 4.2m)$, the corresponding grid on the finest level has a resolution of $(150mm \times 105mm)$.

Our current implementation of the fast oriented distance transform uses 12 discrete orientations, ranging from 0 to π, with the validation gate of 50 pixels. As it computes each orientation separately, they overall require about 0.12 sec/frame for four images, whereas the oriented DT in our previous work [26] is computed in 0.25 sec/frame. Therefore, we really speed up our oriented DT computation with our new algorithm.

Fig. 6 shows qualitative tracking results in a multi-camera environment, with a complex background. In particular, the top row shows foreground edges after



(a) Frame 185



(b) Frame 345

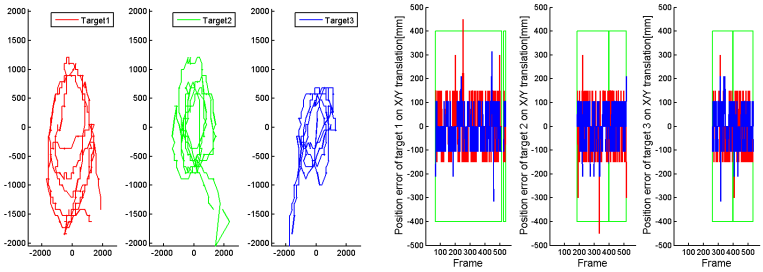
Fig. 6. Performance of 3D people tracking. Shown are edge-based background subtraction, likelihood grids, and the corresponding tracking results, on four camera views.

edge-based background subtraction. Here 30 frames have been used for background learning, where the threshold θ mentioned in Eq. (2) is set to 0.9. The middle row shows likelihood values onto the finest grid, and the bottom row shows the corresponding tracking results after data association, with the projected cylinder silhouettes.

During data association, we keep a confirmation time of 10 frames (which is the maximum value for the consecutive detections counter) for keeping or removing tracks. As can be seen from the results, there are situations with significant occlusion from one or more views. For instance, at frame 345, each two targets are occluded from some views, however, since for the same pairs there are no occlusions from another camera view, all targets are successfully detected, thanks

to the robustness of multi-camera fusion and oriented DT matching. The system also successfully handles targets entering and leaving the scene.

In order to better evaluate the performance of our system, we manually label the ground truth data for our sequences, and compare the results of our tracker, both in terms of position accuracy and robustness of detection. Ground truth trajectories, labeled on the finest grid, are depicted in Fig. 7(a), where we can see the challenges due to targets that keep close most of the time, with mutual interactions and position exchanges.



(a) (b)

Fig. 7. Ground truth evaluation. (a) Ground-truth trajectories, sampled on the discretized grid. (b) Position error on X(red) and Y(blue) in world coordinates.

Fig. 7(b) shows the (X, Y) position errors of our tracking system. Although the above mentioned occlusions and dynamics, for each target the system can basically keep the track. For the temporal track lost case, compared to our previous work [26], it is improved to happens only 2 times per target over the 550 frames of sequence and can recover again very shortly afterwards. This is attributed to the utilization of updated 3-cylinder model. The leaded sub-tracks with different IDs have been also shown in Fig. 7(b) by the green boxes.

Overall these results indicate that, despite the cluttered situation, position errors are considerably low for all people, being most of the time under 100-150mm, that corresponds to one cell of the high-resolution grid. This is because of the local edge-based matching which, despite the simplicity of the model, is more precise with respect to global statistics such as color histograms, or histograms of oriented gradients.

The execution time of the whole tracking procedure is currently 2.8 FPS, on a desktop PC with Intel Core 2 Duo CPU (1.86 GHz), 1GB RAM and an Nvidia GeForce 8600 GT graphic card.

7 Conclusion

In this paper, we presented a novel system for multiple people tracking in a multi-camera environment, using a grid-based tracking by detection methodology. A

template hierarchy is constructed off-line, by partitioning the state space. And frame-by-frame detection is performed by means of hierarchical likelihood grids and clustered on the finest level, followed by data association through the GNN approach. Moreover, edge-based background subtraction has been proposed for foreground segmentation, which is quite robust to illumination changes, together with a speeded-up oriented distance transform, matching the silhouette templates by taking gradient orientations into account, thus significantly reducing the rate of false alarms. Our system initiates, maintains and terminates tracks in a fully automatic way. Experimental results over the video sequences also show that our proposed system deals fairly well with mutual occlusions.

As a future work, this system can be easily extended to include additional features, such as color or motion, also can be scaled to more camera views, as well as being used for tracking different objects, for example 3D indoor tracking of flying *quadrotors*. In addition, the individual components can still be further optimized, both with respect to speed and performance, graphics hardware is possibly need to be exploited. Moreover, we plan to address the issue of heavy occlusions between people, taking place for longer periods. Re-identification after occlusions are going to be done by using more specific features, such as color or texture.

Besides these straightforward improvements, we also plan to test and extend our system to more challenging scenarios, such as outdoor tracking with multiple models (such as people and cars), as well as people tracking on mobile robots, with a non-static background and viewpoint.

References

1. Gavrila, D.M.: Pedestrian Detection from a Moving Vehicle. In: Proc. of European Conference on Computer Vision, pp. 37–49. Dublin, Ireland (2000)
2. Panin, G., Lenz, C., Nair, S., Roth, E., Wojtczyk, M., Friedlhuber, T., Knoll, A.: A Unifying Software Architecture for Model-Based Visual Tracking. In: IS&T/SPIE 20th Annual Symposium of Electronic Imaging. San Jose, CA (2008)
3. Olsen, C.F., Huttenlocher, D.P.: Automatic Target Recognition by Matching Oriented Edge Pixels. IEEE Transactions on Image Processing 6(1), 103–113 (1997)
4. Andriluka, M., Roth, S., Schile, B.: People-Tracking-by-Detection and People-Detection-by-Tracking. In: IEEE Conference on Computer Vision and Pattern Recognition (2008)
5. Wu, B., Nevatia, R.: Detection and Tracking of Multiple, Partially Occluded Humans by Bayesian Combination of Edgelet based Part Detectors. International Journal of Computer Vision 75(2), 247–266 (2007)
6. Stauffer, C., Grimson, W.E.L.: Learning Patterns of Activity using Real-Time Tracking. IEEE Transaction on Pattern Analysis and Machine Intelligence 22(8), 747–757 (2000)
7. Eng, H., Wang, J., Kam, A., Yau, W.: A Bayesian Framework for Robust Human Detection and Occlusion Handling using a Human Shape Model. In: International Conference on Pattern Recognition, pp. 257–260 (2004)
8. Okuma, K., Taleghani, A., De Freitas, N., Little, J., Lowe, D.: A Boosted Particle Filter: Multitarget Detection and Tracking. In: European Conference on Computer Vision (2004)

9. Leibe, B., Schindler, K., Van Gool, L.: Coupled Detection and Trajectory Estimation for Multi-Object Tracking. In: International Conference on Computer Vision (2007)
10. Panin, G.: Model-based Visual Tracking: the OpenTL Framework. Wiley-Blackwell (2011)
11. Bar-Shalom, Y., Li, X.: Multitarget-Multisensor Tracking: Principles and Techniques. YBS Publishing (1995)
12. Bar-Shalom, Y., Fortmann, T.E.: Tracking and Data Association. Academic Press, San Diego (1988)
13. Fortmann, T.E., Bar-Shalom, Y., Scheffe, M.: Sonar Tracking of Multiple Targets using Joint Probabilistic Data Association. IEEE Journal of Oceanic Engineering 8(3), 173–184 (1983)
14. Reid, D.B.: An Algorithm for Tracking Multiple Targets. IEEE Transaction on Automatic Control 24(6), 843–854 (1979)
15. Konstantinova, P., Udvarev, A., Semerdjiev, T.: A Study of a Target Tracking Algorithm using Global Nearest Neighbor Approach. In: Proceeding of International Conference on Computer Systems and Technologies (2003)
16. Burgeois. F., Lasalle, J.C.: An Extension of the Munkres Algorithm for the Assignment Problem to Rectangular Matrices. Communications of the ACM, 802–806 (1971)
17. Fieguth, P., Terzopoulos, D.: Color-based Tracking of Heads and other Mobile Objects at Video Frame Rates. In: Proceedings IEEE Conf. on Computer Vision and Pattern Recognition, pp. 21–27. San Juan, Puerto Rico (1997)
18. Roh, M.C., Kim, T.Y., Park, J., Lee, S.W.: Accurate Object Contour Tracking based on Boundary Edge Selection. Pattern Recognition 40(3), 931–943 (2007)
19. Nguyen, H.T., Worring, M., Van Den Boomgaard, R., Smeulders, A.W.M.: Tracking Nonparameterized Object Contours in Video. IEEE Trans. Image Process 11(9), 1081–1091 (2002)
20. Andriluka, M., Roth, S., Schiele, B.: Monocular 3D Pose Estimation and Tracking by Detection. In: IEEE conference on Computer Vision and Pattern Recognition (2010)
21. Khan. S., Javed, O., Rasheed, Z., Shah, M.,: Human Tracking in Multiple Cameras. In: Proceedings of the 8th IEEE International Conference on Computer Vision, pp. 331–336. Vancouver, Canadam (2001)
22. Li, L., Huang, W., Gu, I.Y.H., Tian, Q.: Foreground Object Detection from Videos Containing Complex Background. In: Proceedings of the 11th ACM International Conference on Multimedia, pp. 2–10 (2003)
23. Gavrila, D.M., Davis, L.S.: 3-D Model-based Tracking of Humans in Action: a Multi-View Approach. In: Proc. IEEE Computer Vision and Pattern Recognition, pp. 73–80. San Francisco (1996)
24. Borgefors, G.: Hierarchical Chamfer Matching: A Parametric Edge Matching Algorithm. IEEE Trans. Pattern Analysis and Machine Intelligence 10(6), 849–865 (1988)
25. Stenger, B., Thayananthan, A., Torr, P.H.S., Cipolla, R.: Model-based Hand Tracking using a Hierarchical Bayesian Filter. IEEE Transactions on Pattern Analysis and Machine Intelligence 28, 1372–1384. IEEE Computer Society, Los Alamitos, CA, USA (2006)
26. Chen, L., Panin, G., Knoll, A.: Multi-camera People Tracking with Hierarchical Likelihood Grids. In: Proceedings of the 6th International Conference on Computer Vision Theory and Applications. INSTICC press, Algarve, Portugal (2011)