REAL-TIME TRACKING WITH AN EMBEDDED 3D CAMERA WITH FPGA PROCESSING

Alessandro Muscoloni and Stefano Mattoccia

Dipartimento di Informatica - Scienza e Ingegneria (DISI)
University of Bologna
Viale Risorgimento 2, 40136 Bologna (Italy)
email: alessandro.muscoloni@studio.unibo.it, stefano.mattoccia@unibo.it

ABSTRACT

People tracking is a crucial component of many intelligent video surveillance systems and recent developments in embedded computing architectures and algorithms allow us to design compact, lightweight and energy efficient systems aimed at tackling this problem. In particular, the advent of cheap RGBD sensing devices enables to exploit depth information as additional cue. In this paper we propose a 3D tracking system aimed to become the basic node of a distributed system for business analytics applications. In the envisioned distributed system each node would consist of a custom stereo camera with on-board FPGA processing coupled with a compact CPU based board. In the basic node proposed in this paper, aimed at raw people tracking within the sensed area of a single device, the custom stereo camera delivers, in real time and with minimal energy requirements, accurate dense depth maps according to state-of-the-art computer vision algorithms. Then, the CPU based system, by processing this information enables reliable 3D people tracking. In our system, deploying the FPGA front-end, the main constraint for real time 3D tracking is concerned with the computing requirement of the CPU based board and, in this paper, we propose a fast and effective node for 3D people tracking algorithm suited for implementation on embedded devices.

Index Terms— 3D, tracking, stereo vision, FPGA, smart camera

1. INTRODUCTION AND RELATED WORK

Intelligent visual surveillance systems are nowadays deployed in countless application scenarios and, thank to the opportunity provided by modern computer vision techniques, video analytics has become very relevant in recent years. Video analytics consists in gathering information from video stream in order to analyze the behavior of people/objects sensed by imaging devices. In particular, behavior analysis for commercial or marketing purposes, often referred to as business analytics, is a specific and relevant field in this area. For instance, being able to analyze where people move in shopping malls or retail stores, might allow managers to improve revenues, reduce queues, count customers, analyze trajectories/*heat maps*, evaluate aimed marketing strategies, control restricted areas, etc. It is worth observing that people tracking, the topic addressed by this paper, is the crucial component in all the outlined cases.

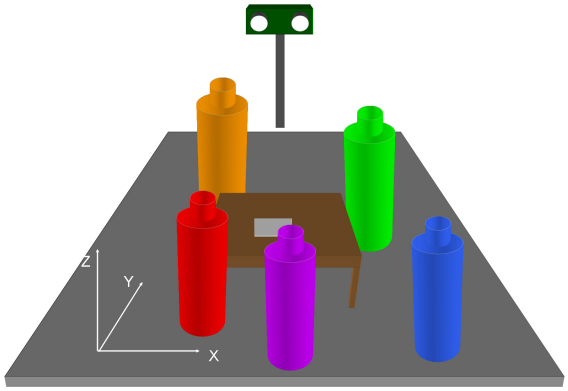

Fig. 1. Typical setup of a 3D people tracking system with downward looking tilted camera.

Other relevant features frequently required in business analytics systems, not addressed in this paper, are concerned with people classification and posture recognition in order to gather further information about people activity in the sensed area. In this field, computer vision is the prominent technique and several approaches aimed at people tracking and counting have been proposed. Some of these are based on cameras mounted at the top of the sensed scene, that is with image planes parallel or almost parallel to the ground plane, to avoid/reduce occlusions. Although solutions based on top mounted cameras may allow to reduce occlusions they have some significant drawbacks compared to solutions based on camera simply tilted with respect to the ground plane and mounted, as frequently occurs in these latter cases, at lower height as depicted in Figure 1. In fact, top mounted solutions in most cases require a more complex infrastructure for placing the cameras and this might be not necessarily available (frequently, in outdoor scenarios). Moreover, with this

setup the sensed area is heavily constrained by the mounting height. For the reasons outlined so far, tilted cameras are the preferred solutions for business analytic applications based on people tracking. Although single 2D cameras and multiple camera setups with overlapping areas are widely used in most practical applications [8], approaches based on dense depth data hold the potential to be more reliable. In this field stereo vision is the most frequently used technology and an early attempt to tackle people tracking using integration of multiple visual modalities - depth, color and face detection - is [2]. However, more effective approaches based on the common idea of ground plane point-cloud projections were proposed in [3, 6, 11]. These methods gather statistics from the foreground points projected on the ground plane and then track people from a *bird-eye* point of view according to different strategies. Common cues used by these methods, as well as by our approach, are *occupancy* and *height* [6] of individuals. Similar cues can be used also for 3D object segmentation/detection as reported in [12]

2. OVERVIEW OF THE 3D TRACKING SYSTEM

The 3D tracking system proposed in this paper is composed of two devices: an embedded 3D camera for dense stereo matching connected to a CPU based board with standard interfaces such as a USB and ethernet. Figure 1 shows the typical setup of the 3D tracking system, with the camera downward facing the sensed area. In this setup, we designed from scratch a 3D camera by mapping algorithms for accurate and real time dense stereo on a low cost FPGA. Regarding the CPU based board we don't have specific constraints and for the experimental results reported in this paper we used a standard notebook. However, embedded ARM boards, such as the ODROID-U3 [5] based on the quad core Exynos processor, are our final target and, for this reason, computational efficiency has been a major constraint in our work.

In the remainder we provide a brief description of our custom 3D camera and a detailed description of our current tracking system implemented on the CPU based board.

3. 3D CAMERA WITH FPGA PROCESSING

Accurate and fast generation of dense depth maps is crucial for 3D tracking and for this purpose we designed an embedded stereo vision camera with FPGA processing. This device, shown in Figure 2, consists of two (monochrome or color) sensors and a processing board with a low cost FPGA belonging to the Xilinx Spartan 6 Family. The two hardware synchronized global shutter sensors have a maximum resolution of 752×480 and a maximum frame rate of 60 fps at this resolution. A more detailed description of the embedded 3D camera can be found in [10].

The stereo video stream acquired by the imaging sensors is entirely processed by an FPGA based board. Into the reconfigurable logic of this device we mapped an entire stereo processing pipeline (i.e., *filtering*, *rectification*, *stereo matching*, *outliers detection*, *subpixel depth interpolation*, etc) based on an appropriate [9] memory efficient version of the SGM algorithm [7]. This camera provides very accurate and reliable dense depth maps and has a power consumption of only 2 Watt processing, at 30+ Hz, stereo pairs at 640×480 pixels. The processing module of the camera, containing the FPGA and the other few components required by our design, is very small (its area is smaller than a business card), lightweight (less than 100g including M12 sensors, lenses and holders) and can be configured with different baselines. Finally, it can be used with different architectures (e.g., Intel and ARM) and OSs (e.g., Windows, Linux and Mac OS).

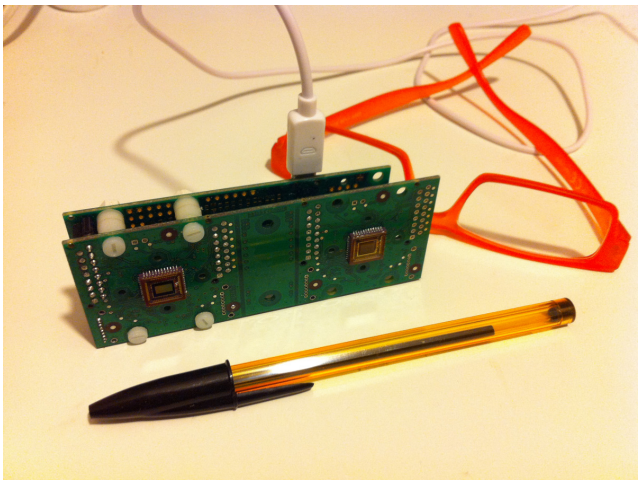

Fig. 2. 3D camera with FPGA processing used for our experimental results. The camera is configured with a baseline of about 6 cm and 640×480 image resolution.

4. SEGMENTATION AND PROJECTION

In this section we describe the people tracking algorithm implemented on a CPU based system that, by processing the dense depth data provided by the stereo camera in real time, allows for reliable people tracking in the observed area. As previously highlighted, computational requirement of the tracking algorithm has been a major constraint in our work in order to enable its deployment on embedded CPU based boards.

4.1. Segmentation

The first step of our tracking system consists in segmenting foreground objects from the background according to the disparity information provided by the 3D camera. For this purpose, at each startup or simply only when the system is configured for the first time, we infer the pose of the camera with respect to a reference system located in the sensed area and we acquire a disparity map without people referred to as D_B.

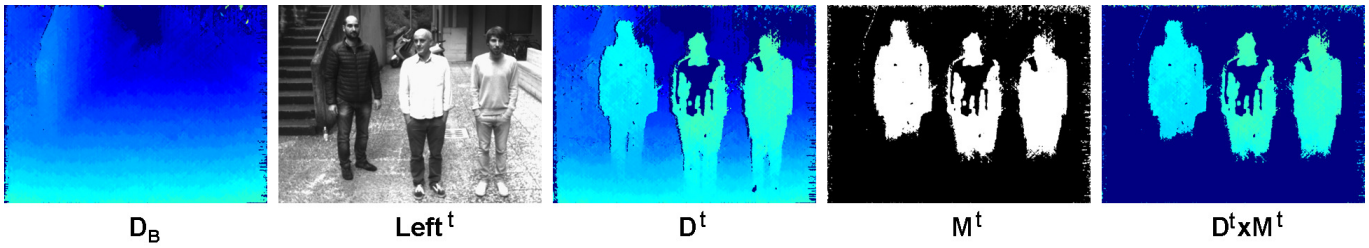

Fig. 3. Foreground segmentation, from left to right: the background disparity map D_B provided by the 3D camera, shown in Figure 2, at startup, the reference image (not used), the disparity map D^t provided by the 3D camera at frame t, the binary map M^t computed according to (1) with th=3, the detected foreground points $D^t \times M^t$

.

For each new frame t the raw disparity map D_B is compared to the current disparity map D^t, by means of a simple background subtraction approach and according to a fixed threshold th, in order to highlight foreground objects. The outcome of this step is a binary map, referred to as M^t, with points at coordinate (x, y) set to 0 or 1 according to[1]

$$M^t(x,y) = \begin{cases} 1, & \text{if } \mid D^t(x,y) - D_B(x,y) \mid > \text{th} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

This method implicitly handles background objects (e.g., furnitures, walls, etc) and, by leveraging the reliable depth information provided by the stereo camera, it is quite robust with respect to different illumination conditions (e.g., shadows, etc) that typically affect 2D based approaches. Nevertheless, more sophisticated (and time consuming) background models based on 3D and/or 2D data would certainly improve the outcome of our raw segmentation. However, this strategy is very fast and provides in most cases only sparse outliers that can be easily dealt with in the projection domain as shown in the remainder. Figure 3 shows the disparity background model D_B computed by the stereo camera at startup/initialization, the reference image of the stereo pair at frame t (not used in our processing pipeline), the disparity map D^t computed by the stereo camera at frame t, the binary map M^t and the segmented disparity map containing foreground points. In this latter map, foreground pixels are the non zero points in M^t with a valid disparity value in D^t (for simplicity, this map will be referred to as $D^t \times M^t$).

4.2. Point cloud projection and filtering

Once computed mask M^t, we project in 3D non zero disparity values of $D^t \times M^t$ according to the calibration parameters of the stereo camera. The resulting point cloud is then processed in order to obtain *top view* (also referred to as *plan-view* or *depth bird-eye view*) statistics of the sensed area according to a reference system determined in the initialization phase.

This strategy, frequently used by 3D-based approaches such as [6], enables to obtain important cues for tracking such as *occupancy* and *height* statistics. The former consists in accumulating on the ground plane, by means of an appropriate normalization factor that takes into account the distance from the camera, the number of 3D points projected within a prefixed area on the ground plane. The latter cue consists in determining the highest value (with respect to the ground plane) among those points projected on the same area of the ground plane. In our experiments, for both cases, the projection area on the ground plane consists of square patches of length 2 cm.

Although other statistics could be inferred from the point cloud, occupancy and height are simple, yet very effective, cues for tracking. Figure 4 show the raw occupancy and height maps computed according to the point cloud concerned with disparity $D^t \times M^t$ shown in Figure 3.

Observing the figure we can notice that the top-view images shown at the top of Figure 4, concerned with raw projection of points $D \times M$ on the ground plane, contain outliers. However, most of these points can be easily removed by applying standard filtering techniques. For this purpose, as shown in the middle of Figure 4, we apply a median filter. The median filtered top-views are further processed by removing in both maps those points with an insufficient amount of occupancy. The resulting height maps is then further processed by keeping only those points compatible with the upper portion of a person (in our experiments we filtered out blobs with height < 70 cm). The outcome of this final processing stage is shown at the bottom of Figure 4; observing the two top-view maps we can clearly notice the blobs concerned with people shown in Figure 3.

Once completed this filtering operations, we stretch the resulting map so as to highlight heads (the most reliable cue during occlusions) and apply the morphological operator *open*.

[1] Although the disparity maps provided by our camera are very dense, some points in the background model and/or in the disparity map at frame t can be however marked as unreliable by the outliers detection algorithm implemented into the FPGA. In these cases, we bypass (1) and we set $M^t(x,y)$ = 1 only when $D_B(x,y)$ is invalid and $D^t(x,y)$ is valid. Otherwise, $M^t(x,y)$ is set to 0.

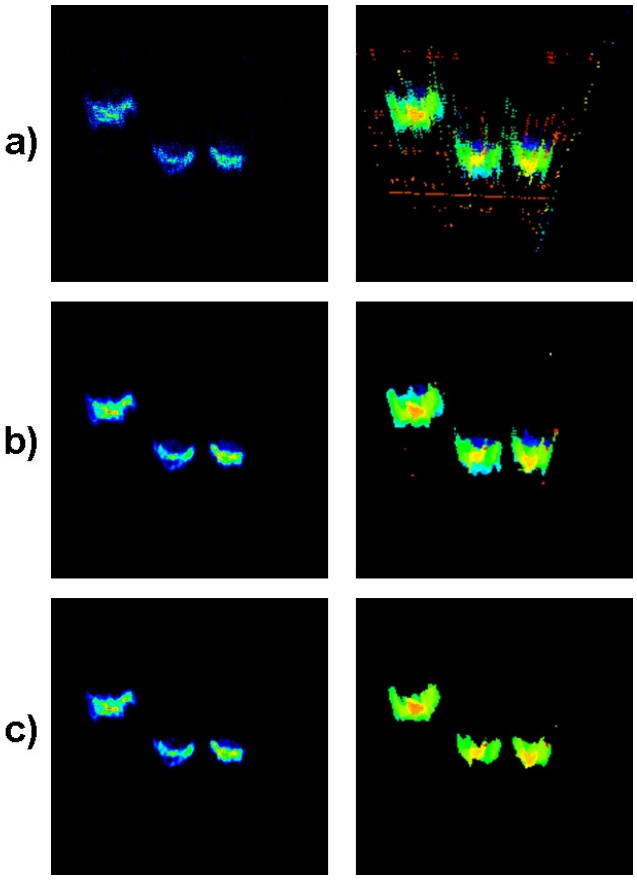

Fig. 4. Top-view projections of map $D \times M$ shown in Figure 3. On the left the occupancy maps and on the right the height maps. (a) Raw maps (b) Median filtered maps (c) Final maps removing smaller blobs and blobs with lower height.

5. PREDICTION AND TRACKING

As first step our tracking algorithm examines, by means of the mean-shift approach driven by a Kalman filtering prediction step, people already tracked in the system and then handles cases concerned with new potential candidates (i.e. people entering the sensed area) and lost people (i.e., people previously tracked and lost in previous frames). For this purpose we keep for each person, among other information, a *state* that can assume the following values: *tracking*, *tracking candidate* and *lost*.

For each frame, once obtained the height map according to the approached outlined in the previous sections, we start by detecting blobs concerned with already tracked people (status = *tracking*, *candidate* or *lost*). For this purpose, for each person with this status, we predict, by means of a Kalman filter assuming a constant velocity model, the position of its blob in the height map at frame t according to the position and speed computed at frame $t-1$. Although the constant velocity model assumed does not necessarily fit

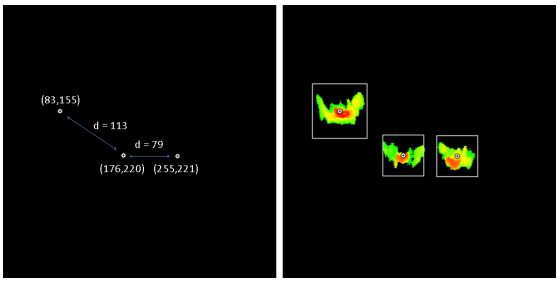

Fig. 5. (Left) Kalman filter prediction for people shown in Figure 3; for each position is reported the distance from the closest person (Right) The size of the kernel window for the mean-shift algorithm changes according to this distance.

with motion models of people entering the sensed area, under different circumstances, the high frame rate allowed by the FPGA stereo camera potentially enables to deal with inaccurate predictions of the Kalman filter. In Figure 5 we can see, on the left, the predicted position for each person shown in Figure 3. In our approach, in order to reduce undesired *blob swapping* events, we set the kernel size for the successive mean-shift tracking algorithm according to the minimum distance from the predicted blob position to any other predicted nearby person. That is, the size of the mean-shift kernel window is proportional to the minimum distance found, with a minimum value compatible with the vertical projection of a person.

Once obtained the predicted position and kernel size for each candidate we carry out data association by means of the mean-shift approach [1]. This algorithm, determines the mode of a discrete density function, encoded in our case by the filtered height map, by iteratively moving a it kernel window in the position of the center of mass. The mean-shift algorithm stops after a fixed number of iterations or, typically, when the outlined procedure converges (that is, once the position of center of mass does not change). In our case, to improve efficiency, the kernel is a uniform rectangular region of size $W = h \times h$ adaptively set according to the minimum distance from nearby predicted candidates.

At frame t, for each tracked person, we update the Kalman filter prediction according to the new measurement, remove from the height map the points overlapping the kernel window and update its tracking status. Concerning lost and candidate people, we keep, repspectively, the number of consecutive frames being not tracked and the number of consecutive frames being tracked. These counters are used to manage tracking status. For instance, a previously tracked person is removed from the system once it has not been tracked (i.e., in *lost* status) for a certain number of consecutive frames.

We highlight new people entered in the sensed area by analyzing the processed height map; in this image we detect height values compatible with those of a person. Once completed this step, in order to handle *lost* people, we try to link

Processing module	ms
Segmentation and point cloud projection	≈ 45
Filtering	≈ 3
Prediction and tracking	≈ 1
Detection of new people	< 1
Handling lost people	≈ 1

Table 1. Measured execution time, with three people in the sensed area, for each module of our approach on a dual core Pentium at 2.1 GHz.

these people with new people detected in the sensed area. In particular, for each person in *lost* status, we consider the closest person entered into the scene within a certain area. The size of this area is increased proportionally to the number of frames being a person in *lost* status. Whether we are able to determine a link between a previously lost person and a new detected person we remove the *lost* person and put the new detected person in the *tracking* status. For the remaining detected people, we set their status to *candidate* and start over. These *candidates* can be promoted to *tracking* status after being tracked for a prefixed number of consecutive frames.

6. EXPERIMENTAL RESULTS

In this section we report experimental results concerned with two sequences processed in real time: one, shown 6 frames in Figure 6, acquired in an indoor faculty corridor and one, shown 6 frames in Figure 7, acquired in an outdoor environment. Both sequences were acquired with our custom stereo camera configured with greyscale sensors (at 640×480 image resolution), a relatively short baseline of about 6 cm and 6 mm M12 lenses. As outlined in Figure 1, the camera is tilted with respect to the ground plane of about 25 degrees and positioned at about 2 m height. With this configuration the sensed area is within a 4 m $\times$ 4 m region on the ground plane. The processing pipeline mapped on the FPGA-based stereo camera includes rectification, our implementation of the SGM algorithm, outliers detection and $\frac{1}{8}$ subpixel interpolation. The camera is connected and self powered by an host computer that, in the reported experimental results, is a notebook equipped with a Pentium dual core T4300 CPU at 2.1 Ghz, 4 GB of RAM and the Windows OS.

For both sequences acquired with this setup we report in figures 6 and 7 the reference image (not used by our current approach) and the outcome of tracking superimposed to the height maps. In this latter images, color encodes the tracking status as follows: purple means *tracking*, red means *lost* and blue means *tracking candidate*.

Table 1 reports, for each main module of our algorithm, the measured average execution time with three people in the sensed area. Although we neither applied specific algorithmic optimizations nor we exploited parallel computing capability, our system runs at almost 20 Hz, a frame rate certainly sufficient to deal with fast walking individuals. The execution time is dominated by point cloud projection, being the execution time linear in the number of non zero points in $D^t \times M^t$. Although this execution time is *somehow* related to the number of people in the scene, it is worth observing that, in the worst case, this time is upper bounded by the processing of the whole image (currently about 150 ms processing all the image pixels in $D^t \times M^t$) independently of the number of people. Finally, we observe that this most time consuming computation would be easily parallelized by exploiting multi core capability today available in most standard as well as embedded CPUs. The other modules of our algorithm have almost negligible execution time. Despite this fact, even in the current unoptimized implementation, the performance of our system scales well with the number of individuals in the scene and its computational requirement seems compatible with our final target architecture based on embedded ARM processors (for instance, the Odroid-U3 [5] is a quad core ARM Cortex A9 at 1.7 GHz with2 GB of RAM).

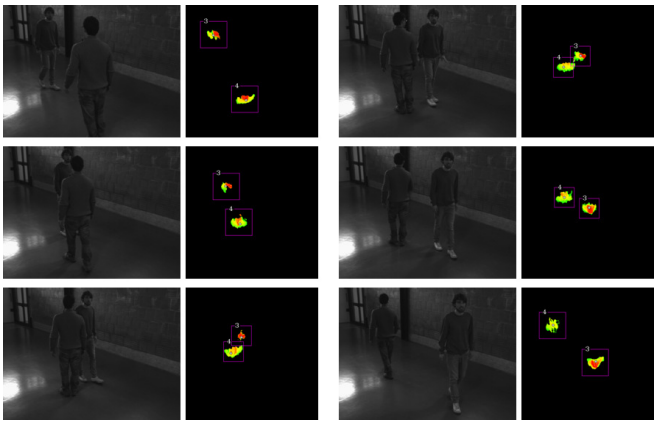

Fig. 6. Indoor sequence (300 frames). The *tracking status* is encoded with colors: purple=tracking, red=lost , blue=tracking candidate.

In figures 6 and 7 we report relevant screen shots concerned with two of the video sequences used for testing our system (300 frames for indoor sequence and 650 frames for outdoor sequence). In the preliminary phase of this work, we also tested the method proposed by Harville [6], however we found that its adaptive template approach for tracking is quite computationally demanding. Moreover, we also found that this approach can easily fail when people get close. Thus, we do not report a comparison with this approach. In both sequences reported in this paper, with camera at very low height (about 2 m), our system correctly handles complete occlusions as well as very close interactions. Observing the top-view height maps reported in the figures we can notice that the disparity maps provided by our stereo camera (reported,

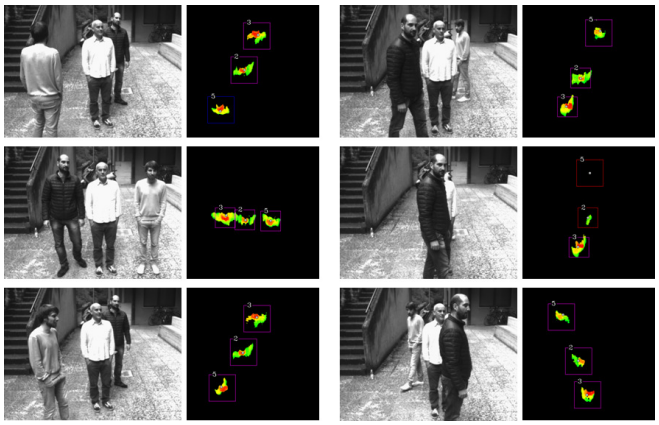

Fig. 7. Outdoor sequence (650 frames). The *tracking status* is encoded with colors: purple=tracking, red=lost , blue=tracking candidate.

due to space constraints, only for the background map D_B and one frame of the outdoor sequence of Figure 3) enable to obtain reliable data for segmentation. In particular the indoor sequence contains specular and uniform surfaces, quite challenging for passive stereo systems. From the same figures we can also notice that the dense depth data provided by the camera, combined with the segmentation and point-cloud projection module proposed, allows to obtain quite reliable cues for tracking even under severe occlusions, as can be noticed in the frames concerned with the outdoor sequence in Figure 7. In fact, observing the rightmost frames of this figure, we can observe that our system can handle the complete occlusion of 2 out of 3 people. From the same frames we can also notice that our system is able to recover from this complete occlusion by correctly managing lost people according to the method previously described.

Although our current system, even with the camera slightly tilted with respect to the ground plane and at very low height, provides reliable tracking information in challenging environments and with severe occlusions sometimes our algorithm fails to handle people *lost* after complete occlusions. For this reason, we plan to improve this phase by including, for each person, additional features (e.g., color histograms) extracted from the rectified reference image provided by the stereo camera.

7. CONCLUSIONS AND FUTURE WORK

In this paper we have proposed a system for fast and effective real-time 3D people tracking. Our system consists of a custom embedded 3D camera with FPGA onboard stereo vision processing and an effective algorithm aimed to CPU implementation. Deploying the FPGA camera as front-end and the algorithm described in this paper, without specific optimizations, currently our system allows for reliable people tracking at about 20 Hz on a Pentium notebook. This makes our proposal suited for porting this algorithm on compact embedded CPU boards based on the ARM architecture using, as for the system proposed in this paper, our stereo camera as front-end for real time depth computation. Excluding this porting, currently in progress, we plan to include in our system fast and effective image based pedestrian detectors (e.g., [4]), in order to further improve tracking effectiveness and to avoid tracking of different objects. Finally, we also plan to include additional low-level features computed from 2D images, currently being not used at all in our algorithm, in order to further improve the management of lost people during occlusions and at handling, in a distributed camera scenario, users across different nodes the surveillance system.

8. REFERENCES

[1] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. PAMI*, 24:603–619, 2002.

[2] T. Darrell, G. Gordon, M. Harville, and J. Woodfill. Integrated person tracking using stereo, color, and pattern detection. *Int. J. Comput. Vision*, 37(2):175–185, June 2000.

[3] Trevor Darrell, David Demirdjian, Neal Checka, and Pedro F. Felzenszwalb. Plan-view trajectory estimation with dense stereo background models. In *ICCV*, pages 628–635, 2001.

[4] P. Dollár, R. Appel, and W. Kienzle. Crosstalk cascades for frame-rate pedestrian detection. In *ECCV*, 2012.

[5] HardKernel. Odroid-u3. http://www.hardkernel.com/main/main.php.

[6] Michael Harville. Stereo person tracking with adaptive plan-view templates of height and occupancy statistics. *Image Vision Comput.*, 22(2):127–142, 2004.

[7] Heiko Hirschmüller. Stereo processing by semiglobal matching and mutual information. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(2):328–341, 2008.

[8] Xi Li, Weiming Hu, Chunhua Shen, Zhongfei Zhang, Anthony Dick, and Anton Van Den Hengel. A survey of appearance models in visual object tracking. *ACM Trans. Intell. Syst. Technol.*, 4(4):58:1–58:48, October 2013.

[9] Stefano Mattoccia. Stereo vision algorithms for fpgas. In *The 9th IEEE Embedded Vision Workshop, CVPR 2013 Workshop*, 2013.

[10] Stefano Mattoccia, Ilario Marchio, and Marco Casadio. A compact 3d camera suited for mobile and embedded

vision applications. In *The Fourth IEEE Workshop on Mobile Vision, CVPR 2014 Workshop*, 2014.

[11] Rafael Muñoz Salinas, Eugenio Aguirre, and Miguel García-Silvente. People detection and tracking using stereo vision and color. *Image Vision Comput.*, 25(6):995–1007, June 2007.

[12] Toru Ubukata, Kenji Terabayashi, Alessandro Moro, and Kazunori Umeda. Multi-object segmentation in a projection plane using subtraction stereo. In *Proceedings of the 2010 20th International Conference on Pattern Recognition*, ICPR '10, pages 3296–3299, 2010.