3D-Sensing Distributed Embedded System for People Tracking and Counting

Andres Burbano*+#, Samir Bouaziz*, Marius Vasiliu*
{andres-alberto.burbano-lopez, samir.bouaziz, marius.vasiliu}@u-psud.fr
*IEF, Université Paris-Saclay ,F91405, Orsay CEDEX, France
+R&D, Shopline, 78220,Viroflay, France

***Abstract*—The present study focuses on the development of an embedded smart camera network dedicated to track and count people in public spaces. In the network, each node is capable of sensing, tracking and counting people while communicating with the adjacent nodes of the network. Each node typically uses a 3D-sensing camera positioned in a *downward-view* but the designed framework can accept other configurations. We present an estimation method for the relative position and orientation of the depth cameras. This system performs background modeling during the calibration process, using a fast and lightweight segmentation algorithm.**

Keywords—people tracking; embedded system; 3D-sensing; background modeling; scene analysis.

I. INTRODUCTION

Industrial security, robotics, urbanism and transportation use computer vision research results to detect, track and count people in private and public spaces. The smart camera concept and networked cameras increase the coverage area of the detection, counting and tracking applications.

Human-sensing is defined by [1] as an assemble of methods to detect presence, to count, to localize, to track and to identify people. This can address a large variety of challenges, widely described in [1] and [2]. Our work uses the same approach, and tries to address several issues: *occlusion* generated by crow overlapping, *person separation* when people are close and/or in physical contact, *sensing noise* generated by sunlight, *scale variation* according to the distance from object to the camera, and finally, *model deformation*, when the person's perspective shifts with respect to the sensor.

Regarding to sensing and embedded technologies, new generations of low cost 3D cameras and boards are now available in the market which keep improving their specifications. Two approaches are evident in the field of people counting and tracking: using classic RGB cameras or smart depth sensors. Depth sensors are not sensitive to reflections or drastic illumination changes, and do not require costly background modeling. These advantages generated a trend in the research community to use 3D sensors to detect, track and count people. Depth sensors are mostly restricted to indoor applications, owing to sunlight interference.

Our work is mainly inspired by two approaches of human detection, tracking and counting methods [3], [4]. The first is an embedded smart camera network that uses a motion histogram from frame differenced images to localize people in color images. The second is a robust 3D sensing system that uses a head–to-shoulders signature for pattern matching after background filtering. Our proposal takes the most relevant concepts from these approaches into the context of a 3D smart camera network, providing efficient algorithms, low energy consumption (resource-constrained) and a non-centralized network system. This allows us to design multi-camera and multi-target tracking systems, leading to a large number of applications, easy installation and scalable nodes.

This paper is organized as follows: Section II describes related works. Section III describes our system design approach. Section IV details smart camera extrinsic calibration methods. People detection approach is found in section V. Section VI explain the tracking and counting process. Finally, the conclusions and future work are presented.

II. RELATED WORK

We introduce the most relevant statements from the literature on people detection, tracking and counting techniques. We focus on: the camera positioning, the detection approaches and the application scenarios.

There is some debate concerning the different positions of the camera, such as *downward-view* or *side-view*. Earlier works [5]–[8] argue the advantages of the downward-view camera positioning: people are observed from overhead [6], avoiding complete occlusion in crowds, notwithstanding there are still occlusion cases. On the other hand, studies like [4], [9] prefer to gather more information about the human body and to use complex methods to remove the background. These works seem to suggest that camera position depends on the usage. Taking this into account, we intend to maximize the coverage area and to minimize the occlusion. This will confirm our initial choice of the *downward-view* because it better fits our system needs. As a result, we use fewer devices and avoid complex computation to detect and separate people.

The approaches to human detection can be mainly grouped into background modeling, object segmentation and pattern matching [1]. We can find several works that use one of these approaches or a combination of them. Background modeling,

978-1-4673-9795-7/15 $31.00 © 2015 IEEE
DOI 10.1109/CSCI.2015.76

using RGB cameras, implies complex algorithms and heavy computation time [10]. Most of these algorithms are interested in the image luminosity changes and avoid motionless people absorption into the background. On the other hand, the 3D sensing approach simplifies the foreground segmentation. In [7], a height threshold allows for fast segmentation, but does not take into account the scenario's irregularities. Contrariwise, Rauter et al. [8] use feature descriptors to avoid the background modeling. However, this requires a manual parameterization of the height of the cameras.

As Teixeira et al. explain in [1], there are two opposed approaches in daily application scenarios: **resource-constrained** vs. **performance-driven**. Some works like [11], [12] are based on resource constraints; however, all of them use color cameras. In the most recent years, several approaches using depth sensors were proposed [7], [13], [14], [8] and [4]. These latter works are based on the performance driven approaches.

In this sense, the advantages of our system approach are: low cost of the system (being more accessible to researchers and industry), low time processing for human detection (using background modeling and human segmentation), a scalable and easy-to-install system, and low energy consumption (environmental impact has become increasingly relevant). Our work intends to introduce the depth sensors mounted in a *downward-view* position in a resource-constrained scenario described in the following section.

III. Smart Camera Platform

Our proposed system is composed of a network of smart cameras. Each node consists of a depth camera, an embedded computer and a network communication unit. Locally, the node detects tracks and counts people in its own scene or FOV (field of view). At the same time, the communication unit sends the tracking information to all its adjacent nodes in the network. This network is built in two architectural levels (graph based, Fig. 1): a communication graph mapped on LAN connections and a coverage graph defined by the geometrical adjacency between locals FOV.

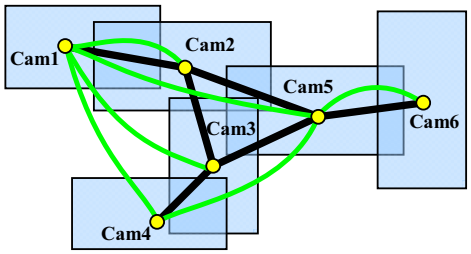

Figure 1. Typical scene covered by 6 cameras: each black rectangle represents a camera FOV and the lines represent the graphs (the coverage graph in black and the communication graph in green).

A. Depth Camera

We use 3D active sensing cameras (like the Asus Xtion Pro or Kinect). These cameras can operate at up to 60 fps (frames per second) delivering (through a USB link) depth images at VGA (640x480) or lower resolutions. Each 12-bit pixel represents the local stereo disparity and can be translated as the distance between the acquired objects and the camera. When there is no sensing information due to occlusion or interference problems, the depth pixel is equal to 0. Typical FOV angles are 58° horizontally and 45° vertically.

Unfortunately, this kind of active sensor has a big disadvantage: sunlight interference restricts its use to only indoors scenarios. However, active 3D sensors can operate in complete darkness. Our system is also compatible with other new passive stereo cameras (like ZED from Stereolabs) but it needs more computing power (not recommended for low-end solutions).

B. Computer Board and Communication Unit

The computer board is made of a Raspberry Pi-2B and an auxiliary proprietary board. The Raspberry-Pi 2 has a Broadcom (BCM2836) quad-core processor (ARM Cortex-A7 MPCore) including a VideoCore IV dual-core GPU, 1GB of SDRAM, SD card class 10, Ethernet networking and a Linux Rasbian distribution OS.

The auxiliary proprietary board, developed by Shopline R&D department, is mainly composed of a Power Over Ethernet (POE) connection and a RTCC (Real-Time Clock and Calendar) integrated circuit. The assembled board is intended to be as cheap as possible and to provide ease of power delivery.

C. Counting Framework Software

We designed a people tracking framework that provides functionality for different kinds of cameras and embedded computers. The cameras supported are RGB, IR and depth cameras. It supports embedded computers running Linux-based operating systems compatibles. The framework also provides remote configuration and communication services. We can setup: one region of interest (ROI), a filtering threshold for the height and counting boundaries (one or more user-defined shapes partitioning the current ROI, in order to count people). During normal operation, the framework acquires depth images, computes *person separation*, tracks and counts people, and sends all relevant information periodically over the network. This framework uses common imaging frameworks (OpenCV, Qt, Open NI) to facilitate data exchange and human computer interaction, but we designed our own optimized image processing routines.

D. Resource-constrained Scenario

Our system was designed taking into account the following constrains:

- A real-time counting system: The smart camera must be able to count and track the people in real time. It must be able to simultaneously send all information to a supervisor system or the adjacent nodes.
- FOV versus installation height: the optimal value of the vertical height of the camera is between 2.5 and 3.5 meters.

- Minimal frame rate acceptance: considering the average human walking speed (1.33m/s) [15] and the mean length of the track for a person in the camera FOV (about 3 meters) the minimum required frame rate is higher than 6 fps.
- The failure-proof functionality: when one node fails, neighbor nodes must take into account the failure and manage the loss of the node remapping the communication graph.

IV. Smart Camera Extrinsic Calibration

The camera calibration variables are divided into intrinsic and extrinsic parameters. As most of the smart cameras have the intrinsic parameters calibrated (focal length, principal point, skew coefficient and distortion), we focus our work on the extrinsic calibration. The extrinsic calibration process consists of robust estimation of the floor plane position giving access to the camera installation height, camera orientation, and the background modeling. As the floor plane and all cameras are fixed, and since the background does not change during short periods of time, the extrinsic calibration process is only used at the initialization stage of the system. Additionally, this operation is too heavy to be done for each new frame. Before starting the computation, each depth pixel $G(x, y)$ is converted to a 3D point $p_i=[x_i, y_i, z_i]$.

A. Camera Self Positioning

In the self-positioning step we extract the floor plane equation by minimizing the square distance between the 3D points cloud and the plane (Fig. 2). In order to accelerate this computation step, we compute the depth histogram and use the range around ($\pm d_0$) the most significant bin B_{max} (assuming that the floor is quasi-horizontal and represents the largest region of the scene; this is true only for the *downward-view*).

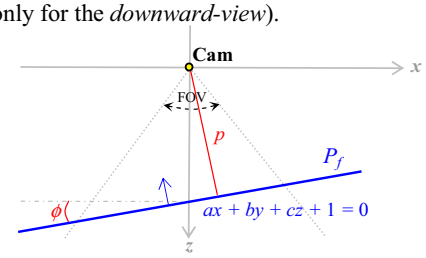

Figure 2. Geometry of the floor plane using a coordinate system with camera CamI as the origin (lateral view from **y** axis).

Let $\Psi = \{p_i | B_{\max} - d_0 \le z_i \le B_{\max} + d_0\}$ be the selected set of N points in the depth image supposed to belong to the floor plane $\boldsymbol{P_F}$ defined by the equation $ax+by+cz+1=0$. If we note $d = \sqrt{a^2 + b^2 + c^2}$ then the sum of the square distances to this plane is:

$$S(\Psi) = \frac{1}{d^2} \sum_{i=1}^{N} (ax_i + by_i + cz_i + 1)^2 \qquad (1)$$

And the best estimation of floor plane parameters is found by:

$$\arg\min_{a,b,c} S(\Psi) = \arg\min_{a,b,c} \frac{1}{d^2} \sum_{i=1}^{N} (ax_i + by_i + cz_i + 1)^2 \qquad (2)$$

If coefficients a and b are too big, the floor plane is not quasi-horizontal and we must reiterate the estimation procedure with a new depth histogram with respect to the normal of the last estimated floor plane.

Finally, this gives us the estimated distance p to the floor (the camera installation height) and the angle $\boldsymbol{\Phi}$ between the camera X axis and the floor plane.

B. Floor Classification

The floor points region $\boldsymbol{R_F}$ is determined as a connected set of depth pixels closer than d_0 to the floor plane $\boldsymbol{P_F}$. The value of d_0 depends on the installation height and the signal to noise ratio of the depth signal, typically between 10 and 15 centimeters.

Fig. 3 shows the floor plane classification using the associated histogram, in two opposite cases: a scene (a) with a simple background, without dynamic objects and another scene (b) with a complex background and some dynamic objects. A simple background can be a floor with very few objects. A complex background has many objects, like tables, chairs, walls, doors, stairs or ramps, etc.

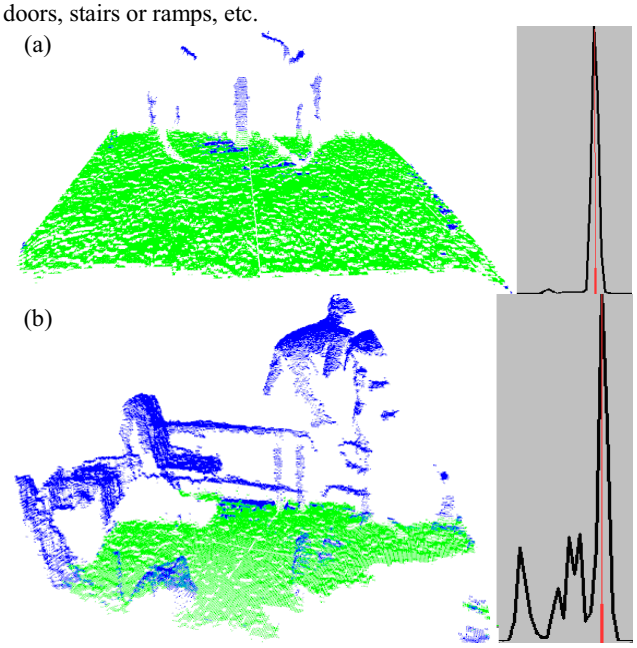

Figure 3. Floor plane classification with the corresponding depth histogram and the bin B_{max} marked in red. Cloud points are acquired from a *downward-view* but are presented here in a lateral view. a) Simple background. b) Complex background.

C. Non-floor objects Classification

The rest of the depth pixels are classified in one or more connected regions $\boldsymbol{R_i}$ topologically defined by the floor region and the image borders. We consider only the large regions connected to the floor. Furthermore, we make the assumption

that most out-of-floor objects can be roughly modeled as plane surfaces. For each region $\boldsymbol{R_i}$ we estimate the local plane $\boldsymbol{Pr^i}$ parameters (a^i, b^i, c^i).

$$\boldsymbol{Pr^i} := \{a^i x + b^i y + c^i z + 1 = 0\} \quad (3)$$

If the region normal is quasi-parallel to the floor plane, this region is identified as a wall or a door.

For the rest of the significant regions, we calculate the relative angle (φ^i) to the floor plane. If this angle is bigger than the building standards maximal angle (ω) [16], we classify them as furniture; otherwise, it becomes a trackable sloped region $\boldsymbol{R_i}$. Regions quasi-parallel to the floor plane can be classified as different floor regions.

Future work will focus on identifying mobile objects with which the humans can interact (trolleys, mobile chairs...).

D. Background modeling

Our system is focused on person tracking and counting in public spaces like shopping malls or supermarkets. A common restriction in this scenario is the large number of objects with which people can interact, such as trolleys. These undesired objects must be discarded from the counting estimation using background modeling. This approach uses the different identified regions to create a virtual filtering surface that takes into account the scenario's irregularities. This virtual surface *Bg(x, y)* is a background threshold helping to distinguish whether a depth pixel belongs to the objects of interest or to the background.

Let $\boldsymbol{L}$ be the set of regions that belong to the scene but are not part of the floor. Let $\boldsymbol{L^t} \subseteq \boldsymbol{L}$ be the set of trackable regions. Let $\boldsymbol{R_i}$ be a trackable region iff $\boldsymbol{R_i} \in \boldsymbol{L^t}$, $\boldsymbol{R_i}$ is connected with $\boldsymbol{R_F}$ and $|\varphi^i| \leq \omega$.

The filtering surface is a patchwork composed of local filtering planes $\boldsymbol{P_t^i}$ for each trackable region $\boldsymbol{R_i}$. Local planes are obtained by raising the estimated plane $\boldsymbol{P^i}$ of local region by a threshold of height t_d as in [7].

$$\boldsymbol{P_t^i} := \{a^i x + b^i y + c^i (z + t_d) + 1 = 0\} := \{a_t{}^i x + b_t{}^i y + c_t{}^i z + 1 = 0\} \quad (4)$$

Then, the background threshold is composed by:

$$Bg(x,y) = \begin{cases} \dfrac{-1}{a_t{}^{Ir(x,y)}\left(\dfrac{x - cc_x}{f_x}\right) + b_t{}^{Ir(x,y)}\left(\dfrac{y - cc_y}{f_y}\right) + c_t{}^{Ir(x,y)}} & G(x,y) > 0 \wedge \\ & Ir(x,y) \neq 0 \\ t_d & otherwise \end{cases} \quad (5)$$

Where *Ir(x, y)* is a labeling function that returns for each depth pixel *G(x, y),* the index *i* of $\boldsymbol{R_i} \in \boldsymbol{L^t}$ in order to get the plane estimated values (a_t^i, b_t^i, c_t^i) or 0 otherwise. The Fig. 4 explains the raising procedure viewed from the $\boldsymbol{y}$ axis.

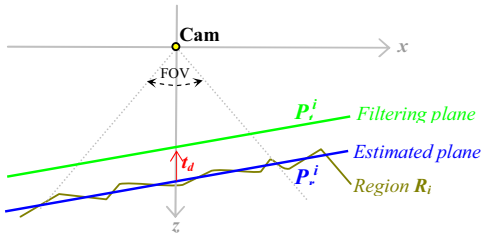

Figure 4. Geometric Representation of the background model,where the green line represents $\boldsymbol{P_t^i}$ and the blue line represents $\boldsymbol{P_r^i}$ (lateral view from $\boldsymbol{y}$ axis).

V. Person-Detection Approach

The people detection process is divided into 3 steps executed for each new depth image (Fig. 5). First, a fast detection of objects of interest using the foreground blob detection method; where we filter and label the blobs. Second is the blob segmentation, where we analyze each blob in detail. Third, graph structure creation where we characterize the people.

Figure 5. Row depth image with several obstacles and two people in contact.

A. Foreground Blobs detection

For the image filtering we use the proposed filtering method in [7], changing the filtering threshold for the background modeling explained in section (IV.D) in order to obtain the foreground image *Fi*:

$$Fi(x,y) = \begin{cases} G(x,y) & G(x,y) \geq Bg(x,y) \\ 0 & otherwise \end{cases} \quad (6)$$

After finding our pixels of interest we continue labeling the connected pixels using the method proposed in [17]. We use an auxiliary table of components [18] for each object where we save important information further use, like barycenter, number of pixels, maximum (*MaxVal*) and minimum (*MinVal*) height from the floor, average height and the bounding box. This requires just a single run, saving calculation time. For computation time purposes, we only apply the next step to blobs bigger than 150% of the human mean size and discard the blobs smaller than ≈3% (depending on the installation height the percentage vary between 2.3 and 3.5).

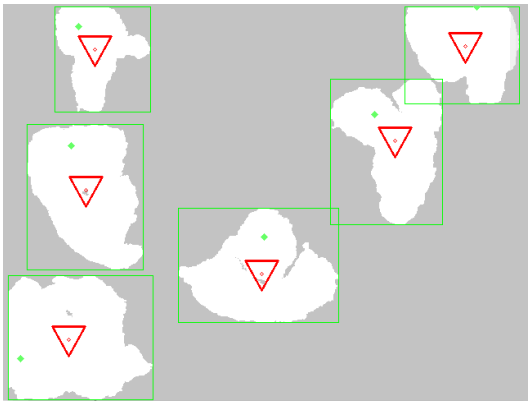

Figure 6. Foreground blobs detected and labeled.

B. Blobs segmentation

The blobs segmentation process consists into two steps. The first one is called level segmentation where each blob is divided in horizontal slices with constant thickness t_c, grouping the pixels in a top-down order. It starts at the maximum value of the blob, *MaxVal*, up to *MinVal*. In consequence, the different pixels are grouped in levels (slices), allowing us to differentiate the head from the shoulders. Now that the pixels are segmented by height, the second step uses the same labeling method than the one described above, to obtain the labeling table. In Fig. 7 we can see the input blob (7a) and the result of the *level segmentation* (7b).

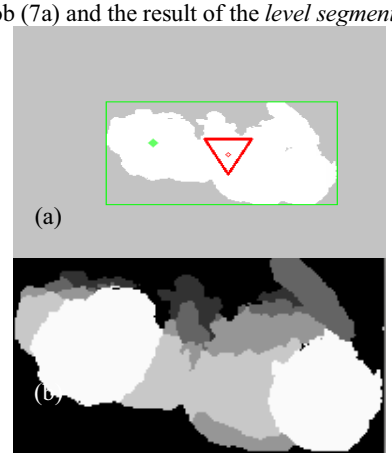

Figure 7. Two people in contact: a) Depth image filtered and labeled. b) Blobs segmented by levels.

C. Graph structure creation

The graph structure is a bi-directional graph called the *search graph* (SG) where every *node* represents a region and every *edge* represents a connection between two regions. We reused the labels table of the blob segmentation and we added a 4-connected property: if one of the pixels in the 4-connectivity does not belong to the same label; we create a new *edge* in the graph. An *edge* has a positive value when the connected node is in a lower *level*, otherwise it is negative. As a result we have a hierarchical graph with respect to the height.

The advantage of representing a human target as a graph is that it gives us the ability to use powerful data representation for use in graph search methods. For simplicity we only take into account 4 layers down from the top. The SG is represented by a matrix of size $card(L^l)$. The matrix values are calculated with the function $SG(r_1,r_2)=level(r_1)-level(r_2)$. In order to find the heads, we look for the nodes with only positive connection values. In other words, only have lower children nodes like the nodes 3 and 6 in the Fig. 8.

Table I shows the resulting connection matrix of the *SG* of Fig. 8b using $SG(r_1,r_2)$. To find the heads, it suffices to find the nodes where all the children are at least one level under. Rows 3 and 6 are these types of nodes in the example table I.

VI. Tracking and Counting

In order to have a better *time-spatial* relation of the tracked objects through the frame sequence, we use the model update [2]. Every time an object is tracked, in the next frame the SG is updated with the current frame information. This model update also serves to solve tracking correlation problems when the algorithm has more than one option to choose.

For the counting algorithm, the system was configured with the simplest counting boundary: two segments modeling an entry and an exit, respectively. These divide the scene into 3 different parts. Instead of calculating whether the full trajectory crosses both lines, we mark the entry zone in the object and we only increase the counters if the actual position of the object is the opposite of the marked initial zone.

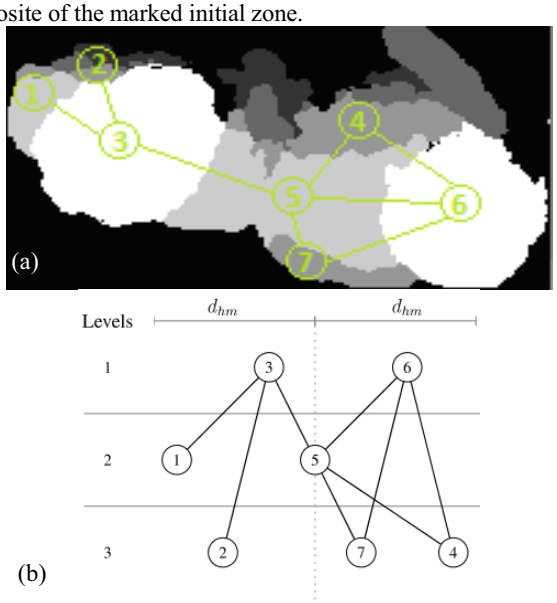

Figure 8. The SG structure representation, where a) the blob is segmented with the top levels nodes of the SG and b) the resulting SG of the segmentation.

TABLE I. SEARCH GRAPH Connections

Node	1	2	3	4	5	6	7
1	0	0	-1	0	0	0	0
2	0	0	-2	0	0	0	0
3	***1***	***2***	***0***	***0***	***1***	***0***	***0***
4	0	0	0	0	-1	-2	0
5	0	0	-1	1	0	-1	1
6	***0***	***0***	***0***	***2***	***1***	***0***	***2***
7	0	0	0	0	-1	-2	0

[a] This table describes the connections between the nodes. Rows 3 and 6 in bold and italic are the nodes that have only connected regions in lower levels; therefore their values are all positives.

VII. CONCLUSIONS AND FUTURE WORK

We created a distributed system for people tracking and counting that uses embedded nodes with a smart depth camera in a *resource-constrained* scenario.

In our approach, the use of a depth sensor allows us to reduce *sensing noise*. Installing the system in a *downward-view* minimize *occlusion* and maximize the coverage. The background model saves calculation time and takes into account irregular trackable surfaces. The *head-to-shoulders signature* represented in a *search graph* introduces a lightweight structure to *separate people*, to send the SG to adjacent nodes and to manage the *model deformation* and *scale variation*. The results are: the installation height estimation has ± 2 cm of error in comparison to the ground truth (even in the complex scenario Fig. 3b).The process runs at 30 fps at VGA resolution in a x64 processor vs 10 fps in the ARM processor, and 30 fps at resolution of 320 x 280 pixels in both architectures (working in lower resolutions decrease the accuracy). Finally the process has an accuracy around 95%. This system is currently industrialized and sold by Shopline Electronic.

This kind of approach can be implemented on low-end embedded systems working in real-time with smart 3D cameras. The whole system is highly reliable, remote-configurable and easy to set up.

Our future research will follow three directions: we will propose new networked camera functionality like the multi-camera network protocol implementation, multi-camera calibration and a fail-proof model. The second direction concerns deeper human characterization by exploring the *search graph* structures related to the human body morphology, in order to get interesting information like the age and the sex. Finally, the third direction concerns improvements to the tracking algorithm in certain particular cases caused by occlusion, where it is not possible to segment two or more humans. In these cases the algorithm might be able to merge/separate multiple person trajectories as proposed in [4], using the spatio-temporal tracking data.

REFERENCES

[1] T. Teixeira, G. Dublon, and A. Savvides, "A survey of human-sensing: Methods for detecting presence, count, location, track, and identity," *ACM Comput. Surv.*, vol. 5, pp. 427–450, 2010.

[2] Y. Wu, J. Lim, and M. Yang, "Object Tracking Benchmark," *Pattern Anal. Mach. Intell. IEEE Trans. On*, vol. 37, no. 9, pp. 1834–1848, Sep. 2015.

[3] T. Teixeira, D. Jung, and A. Savvides, "Tasking Networked CCTV Cameras and Mobile Phones to Identify and Localize Multiple People," in *Procs. 12th ACM Int. Conf. on Ubiquitous Computing*, New York, NY, USA, 2010, pp. 213–222.

[4] N. Kirchner, A. Alempijevic, X. Dai, P. Plöger, and R. K. Venkat, "A robust people detection, tracking, and counting system," in *International Conference on Robotics and Automation*, 2014.

[5] K. Terada, D. Yoshida, S. Oe, and J. Yamaguchi, "A method of counting the passing people by using the stereo images," in *Image Processing, Procs. Int.l Conf. on*, 1999, vol. 2, pp. 338–342.

[6] T.-H. Chen, "An automatic bi-directional passing-people counting method based on color image processing," in *Procs. 37th Inter Carnahan Conf. on Security Technology*, 2003, pp. 200–207.

[7] Z. Qiuyu, T. Li, J. Yiping, and D. Wei-jun, "A novel approach of counting people based on stereovision and DSP," in *The 2nd Inter. Conf. on Computer and Automation Engineering (ICCAE)*, 2010, vol. 1, pp. 81–84.

[8] M. Rauter, "Reliable human detection and tracking in top-view depth images," in *Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 529–534.

[9] H. Fu, H. Ma, and H. Xiao, "Real-time accurate crowd counting based on RGB-D information," in *19th IEEE International Conference on Image Processing (ICIP)*, 2012, pp. 2685–2688.

[10] C.-H. Chen, T.-Y. Chen, D.-J. Wang, and T.-J. Chen, "A cost-effective people-counter for a crowd of moving people based on two-stage segmentation," *J. Inf. Hiding Multimed. Signal Process.*, vol. 3, no. 1, pp. 12–25, 2012.

[11] T. Teixeira and A. Savvides, "Lightweight People Counting and Localizing for Easily Deployable Indoors WSNs," *IEEE J. Sel. Top. Signal Process.*, vol. 2, no. 4, pp. 493–502, Aug. 2008.

[12] Youlu Wang, S. Velipasalar, and M. Casares, "Cooperative Object Tracking and Composite Event Detection With Wireless Embedded Smart Cameras," *IEEE Trans. Image Process.*, vol. 19, no. 10, pp. 2614–2633, Oct. 2010.

[13] H. Fu, H. Ma, and L. Liu, "Robust Human Detection with Low Energy Consumption in Visual Sensor Network," *Seventh International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, 2011, pp. 91–97.

[14] J. Liu, Y. Liu, Y. Cui, and Y. Q. Chen, "Real-time human detection and tracking in complex environments using single RGBD camera," *20th IEEE Int. Conf. on Image Processing (ICIP)*, 2013, pp. 3088–3092.

[15] R. C. Browning, E. A. Baker, J. A. Herron, and R. Kram, "Effects of obesity and sex on the energetic cost and preferred speed of walking," *J. Appl. Physiol.*, vol. 100, no. 2, pp. 390–398, Feb. 2006.

[16] M. P. Gibbens, *Caldag 2013: An Interpretive Manual and Checklist*. International Code Council, 2013, pp. 138–140.

[17] L. Di Stefano and A. Bulgarelli, "A simple and efficient connected components labeling algorithm," in *Image Analysis and Proc., Int. Conf. on*, 1999, pp. 322–327.

[18] A. Rakhmadi, N. Z. Othman, A. Bade, M. S. Rahim, and I. M. Amin, "Connected component labeling using components neighbors-scan labeling approach," *J. Comput. Sci.*, vol. 6, no. 10, 2010, pp. 1099.