Reliable Human Detection and Tracking in Top-View Depth Images

Michael Rauter
Austrian Institute of Technology
Donau-City-Strasse 1, 1220 Vienna, Austria
michael.rauter@ait.ac.at

Abstract

The paper presents a method for human detection and tracking in depth images captured by a top-view camera system. We introduce a new feature descriptor which outperforms state-of-the-art features like Simplified Local Ternary Patterns in the given scenario. We use this feature descriptor to train a head-shoulder detector using a discriminative class scheme. A separate processing step ensures that only a minimal but sufficient number of head-shoulder candidates is evaluated. This contributes to an excellent runtime performance. A final tracking step reliably propagates detections in time and provides stable tracking results. The quality of the presented method allows us to recognize many challenging situations with humans tailgating and piggybacking.

1. Introduction

Detection of human individuals in video data is an essential need in the field of video surveillance. One is often interested in derived information from these detections, such as human tracks or the count of humans in the scene. Multiple humans in the observed scene can lead to challenging problems. Segmentation of human individuals close to each other, potentially overlapping each other is one example. This problem is often encountered when a traditional camera setup is used under a 2D side view. In this context robust human detection and separation becomes an extremely difficult task especially when dealing with crowded scenes.
Illumination changes in scenes are another burden when trying to develop a reliable human detection algorithm. Especially color data captured by traditional color cameras is sensitive to these illumination changes since the pixel values can change drastically in consecutive frames over a short period of time. Furthermore, shadows and reflections can lead to spurious results of a detection algorithm.
Human detection and counting based on range data is becoming more and more popular due to the benefits over traditional approaches based on color data. Stereo camera systems provide solutions for shortcomings of traditional methods like sensitivity to illumination conditions.
When the camera is mounted above the observed humans and is pointing straight down the problem of humans occluding each other can be decreased or even eliminated. Nevertheless, certain tasks remain a challenge like humans walking close to each other (piggybacking, tailgating).
This paper presents a new method for human detection and tracking in top-view depth images. We want to point out three important benefits of our method: (1) In contrast to many other approaches our method does not rely on background modeling for foreground/background segmentation. Therefore it does not suffer from typical problems related to background modeling, e.g. foreground objects becoming part of the background after a certain period of time or, sensitivity to short-term illumination/lighting changes and invalid foreground segmentation arising from it. (2) We present a new feature descriptor to maximize the detection performance of a discriminatively trained head-shoulder classifier. (3) We present how we minimize the number of input head candidates (potential heads that need to be tested) for the classifier to decrease processing time drastically.
A discussion of related work follows in Section 2. In Section 3 we describe the method developed. Next, we present experiments and results in Section 4. Finally we give a conclusion of our paper in Section 5.

2. Related Work

Several methods have been proposed which employ depth information and use a traditional side-view camera scenario. In [7] a mean and standard deviation background model of the background image is maintained, and background subtraction is applied to detect foreground blobs. Depth information of each foreground blob is used to estimate the location of humans in the scene, though head detection is done without stereo information by applying a partial-ellipse fitting algorithm. In [6] a stereo-based background subtraction algorithm is used to segment the image into different disparity layers which are used in a contour fit-

ting step. Finally a tracking step is performed. In [5] stereo-based human head detection is done with scale-adaptive filters based on an elliptical shape approximating the head contour. They suppress spurious clues and localize heads in the likelihood map with a mean-shift algorithm. In another work [9] a method for human detection is presented which uses histogram of oriented depths features to perform a scale-space search and a combination of these features and color data is proposed. In [12] Histogram of Depth Difference is proposed as a new feature descriptor. The detection window is partitioned into grids and blocks and local descriptors for depth differences in the x- and y-direction are aggregated into a histogram. They use this descriptor to train a support vector machine classifier to detect pedestrians in depth images. In [14] another feature descriptor is presented, the Simplified Local Ternary Pattern. Depth differences in x- and y-direction are quantized into 0,+1, and -1 indicating their difference relationship and leading to 9 different observable patterns. The detection window is partitioned into non-overlapping blocks and a histogram is build from the blocks.

Other approaches use a setup where the camera is mounted above the humans observed and pointing straight down at the scene. In [1] a 3D volume of interest in head height is investigated to locate heads. The depth image is resampled from a vertical projection into a occupancy map to remove perspective distortion. The occupancy map is used to compute the locations of humans with a gaussian mixture model. In [13] the height map is segmented into different height intervals resulting in binary images for each level containing only regions with a height belonging to the corresponding interval. Morphological operations as openings with circular structuring elements are used to locate the human heads. Tracking of these heads is done in a tracking step and a Kalman filter for predicting the head positions is employed. In [10] an adaptive background model is used to extract foreground regions. On these foreground regions a spherical crust template is matched. They determine the number of heads and their position and height in a foreground blob. In a blob separation step they split the foreground blobs that contain more than one head. Eventually detections are fed into a tracker based on a set of Kalman filters.

3. Method

We propose a method for human detection, tracking and counting consisting of several algorithmic steps. In the first step we search the depth map for local maxima 3.1. These are used as seed points for a head localization algorithm in the next step 3.2. The centers of the head candidates are found in this step and handed over to the classification-based head detector 3.3. The head detector predicts if the head candidate is a valid head. In a final step a tracking algorithm aggregates the detections over time and computes trajectories for every head 3.4.

Note that our algorithm does not depend on background modeling for foreground/background segmentation. Therefore it does not suffer from its typical shortcomings, e.g. a person that stands still for a while eventually gets integrated into the background (and is not detected anymore). Furthermore abrupt short-term illumination/lighting changes can lead to incorrect foreground segmentation.

The foundation for the method presented is the top-view depth image of the scene provided by a 3D stereo camera system. The 8-bit depth image contains pixels of smaller depth values for near objects and higher values for objects more distant to the camera. Invalid pixels are encoded with a zero indicating that there is no valid depth information available for this pixel position. In our scenario the mount height of the camera is known beforehand. So we can easily determine the height of an object in world coordinates. In an additional step we invert depth values of the depth map. Therefore depth values of objects on the floor start with small values and become larger for nearer objects (pixels with zero depth value indicating invalid pixels are kept unchanged).

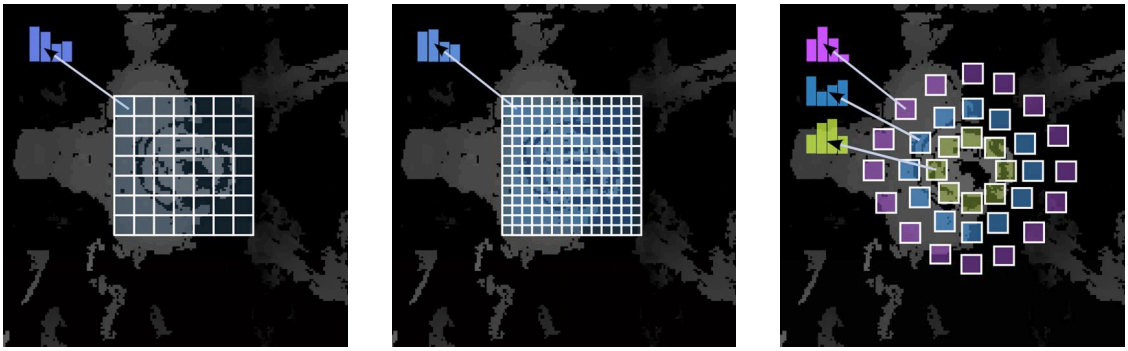
3.1. Local maxima search

Aim of this algorithmic step is to create a sufficient number of initial head candidates for the subsequent step, the head localization algorithm. We achieve this by computing local maxima in the depth image. The size of the search window for the maxima search was set to the minimum expected head diameter (head diameter size of a head near the floor).

Equation 2 in Section 3.3 is used to compute object dimensions (in pixels) for a given object of well-known size and distance from the camera.

We extend the 2D-neighborhood scan-line algorithm for Non-Maxima Suppression from [8] to search the depth image for local maxima. Our extensions to the original algorithm comprise three aspects, (1) we enhance the algorithm to support non-square search windows, (2) we allow maxima plateaus to generate local maxima for the plateau in the search window, (3) we allow multiple local maxima with the exact same value in the search window to generate local maxima.

Note that plateaus or multiple maxima with the exact same depth value can and will indeed occur in practice. This is due to the fact that we use 8-bit depth images and several identical depth values in local neighborhood are not unlikely. Thus the presented extensions to the original algorithm are vital to prevent heads from being suppressed by accident.



(a) regular non-overlapping grids (b) overlapping grids (c) rings of grids (separate histograms)

Figure 1. Different configurations of grids

3.2. Head localization algorithm

Candidate points (seed points) created by the local maxima search are likely to differ from the true center points of the head candidates. This true center of the head is exactly what we are interested in since we want to use it as the input for the head detector later on. The purpose of this algorithmic step is thus to localize the centers of the heads.
We use a gradient climbing algorithm similar to the Mean-Shift algorithm [3]. We take into account invalid depth values in the computation of the shift vector. A search window of the size of the expected diameter of a head is used. This size can be computed from the seed point's depth value. By performing this gradient climbing algorithm for all seed points we get a list of final head candidates. Different initial head candidate points can result in final head candidate points with the same position. We take account for this by eliminating multiples in this list.

3.3. Head detection

We propose a new depth feature descriptor which uses a reference value to compute depth differences.
The motivation for our feature descriptor is the fast and accurate computation of mean depth values within a rectangular pixel area in the depth image. Note that the depth image may contain invalid depth values marked as zero values. We have to take invalid pixels into account in the calculation (see Equation 1). We will further call this rectangular pixel area a block. The computation of mean depth values within such a block is done with:

$$\overline{d_B} = \frac{\sum\limits_{x \in B} i_D}{\left(\|B\| - \sum\limits_{x \in B} i_I \right)} \tag{1}$$

where i_D is the depth image, i_I is a map of zeros and ones where a one indicates invalid depth values, and B is the block of pixels. We need to consider blocks of pixels instead of individual pixels since we want to use a scale-adaptive detection window. We use integral images [11] for i_D and i_I to efficiently compute the corresponding sum over the values of the block. Invalid depth pixels are taken into account to ensure correctness of the result. The value of invalid depth pixels is zero and this would decrease the mean block pixel sum when the block sum is divided by the full number of pixels in the block area. So the amount of invalid pixels is computed and subtracted from the denominator.
In order to create the description of local depth neighborhood we use a reference block (centered at the head location) to compute depth differences between other blocks and this reference block. A histogram of these depth difference values is created and the vector of histogram values of the histogram bins is the input for the SVM.
We tested several sets of configurations of blocks. Our tests included regular grids of blocks of different sizes both non-overlapping (see Figure 1(a)) and overlapping (see Figure 1(b)) as well as manually placed blocks over the detection window. We now present the manual placement of blocks and we want to point out that this presented configuration of blocks results in an invariance to rotation. We define several sets of blocks in a circular pattern around the reference block (see Figure 1(c)). Each ring of blocks has a different radius from the center of the reference block. We create separate histograms of these rings of blocks and concatenate all histogram bins into a vector. This vector is the input for the SVM.
The size of the blocks is derived from the detection window size. The detection window itself is scale-adaptive as explained next.
The size of the detection window in pixels can be computed with

$$s_w = \frac{f}{d} \cdot s_r \tag{2}$$

where f is the focal length in pixels, d is the expected distance of the object and s_r is the extend of the object in pixels perpendicular to the optical axis of the camera.

We use support vector machine (SVM) learning and prediction to classify head candidates. The features presented above were used to form data vectors for the svm training and classification. We use the LIBSVM [2] library with the radial basis function kernel. We do a grid-search to optimize parameters with cross validation as suggested by the LIBSVM documentation.
As one can see in Section 4, in our required scenario our feature descriptor outperforms the Simplified Local Ternary Patterns [14] feature descriptor, which is claimed in their work to outperform other descriptors from [9], [12].

3.4. Tracking algorithm

The final algorithmic step is a tracking algorithm. The tracking algorithm tries to find an association of detections in the current frame and already tracked object (tracks) from previous frames. The resulting trajectories allow to derive information from them and one can see the path of movement of the tracked persons. Furthermore with tracking it is possible to suppress single false alarms of the detector.
Our solution to the tracking problem is only an approximation to the optimal solution since we had the requirement of runtime performance in mind.
An association distance cost matrix $C_{d,t}$ is computed for all possible detections d and tracks t. We use the squared Euclidean distance to compute the distance costs. A maximum allowed distance for associations is defined by a threshold. Association distances larger than this distance are penalized, and thus should prevent this association to be selected for the final association configuration. We try to find an optimized association configuration of detections and tracks by computing costs for three different association rules and pick the configuration with minimal costs. The three association rules are: (1) nearest neighbor association (detections are sorted by the minimal distance and associated in this order), (2) tracks with minimal number of allowed detections for association (which are an allowed candidate for this track) and (3) detections with minimal number of allowed tracks for association.
In the vast majority of cases the nearest neighbor association configuration is picked by the algorithm. There are though difficult cases where (2) and (3) are indeed the better choice. Especially when dealing with low frame-rates and people move fast and thus their inter-frame displacement is large it happens that the nearest neighbor rule picks a suboptimal configuration. We maintain a list of tracks (both potential and valid). In every frame the best association configuration of detections and tracks is computed. Existing tracks are updated with the new position of the associated detection (the old position is also saved in their track history). Unassociated tracks are marked as tracks with a failed association and their position is not updated. When a track could not be associated for a predefined number of frames we erase it from the list of tracks. Detections that were not associated to an existing track in the current frame will be used to start a new track. A new track is by default a potential track (invalid). A potential track is required to be detected for a predefined number of frames to become a valid track. This way, occasional false alarms of the detector will not become valid tracks and thus decrease the overall performance of the method.

4. Experiments and Results

We carried out a comparison of the detection performance of the SVM classifier based detector employing our features as well as Simplified Local Ternary Patterns (SLTP) features. In [14] SLTP features are presented and evaluated. It is shown that SLTP outperforms Histogram of Oriented Depths (HOG) [4] features proposed in [9] and Histogram of Depth Difference (HDD) features invented in [12]. Like our features HDD features use depth differences. But instead of using a depth reference value only local depth differences are computed. As we will show our features outperform SLTP features in our scenario, which again are superior to HDD features as claimed in [14].
We did our tests on a learning dataset of 4212 positive and 14040 negative samples and a test dataset of 5184 positive and 7344 negative samples. Samples were annotated manually and rotated automatically in steps of 10° rotations to generate a larger number of samples and to increase variation in the data sets. In our tests we found our features to perform better than SLTP features (see Table 1). While the true positive detection rate is pretty good for all variants our approach is superior when it comes to true negative detection rate.

detection performance of SLTP features			
regular grid of blocks		grid of blocks (overlapping)	
true positive rate	0.91	true positive rate	0.91
true negative rate	0.36	true negative rate	0.54
detection performance of our features			
regular grid of blocks		sets of rings of blocks	
true positive rate	0.90	true positive rate	0.93
true negative rate	0.95	true negative rate	0.99

Table 1. Feature performance comparison

We implemented our algorithm in C++. We used the *Intel Integrated Performance Primitives* for fast image operations. We used LIBSVM [2] for support vector machine training and classification. The runtime performance of our algorithm is shown in Table 2. The test system consisted of an *Intel Xeon* CPU with 4 physical and 4 virtual cores @ 2.93 GHz, 12 GB RAM running on Windows7 64-bit ma-

chine. We compiled both 32-bit and 64-bit versions of the algorithm. The 64-bit version is slightly faster than the 32-bit version. This was expected as the operating system is a native 64-bit system. We therefore only give timing results for the 64-bit implementation. The algorithm is only running on one processor core at the moment.
The head hypothesis generation takes up the largest amount of time. This is expected since it involves both the maxima search as well as the gradient climbing algorithm. Especially the maxima search involves scanning the whole image and thus is computationally intensive. Computation time of the SVM head classification for all head candidates is moderate due to the fact that the number of head candidates was drastically reduced in the head hypothesis generation step. Thus the processing time for this step is about 1 millisecond. The conversion of depth values as mentioned in Section 3 takes up 0.5 milliseconds. As one can see from table 2 the remaining steps (feature computation and the tracking) are negligible.
The overall processing time with less than 3.5 milliseconds is promising.

runtime performance of the algorithm

algorithmic step	runtime
depth values conversion	0.52 msec
head hypothesis generation	1.77 msec
depth difference histogram computation	0.04 msec
svm-based head classification	1.09 msec
association/tracking	0.01 msec
total	3.43 msec

algorithm runtime (average of 1000 frames)
depth image dimensions: 608×328 pixels
test system: *Intel Xeon* CPU with 4 physical and 4 virtual cores @ 2.93 GHz, 12 GB RAM running on Windows7 64-bit

Table 2. Average runtime performance of the algorithm

The algorithm was incorporated in an automatic border control system where person separation and counting are required. In Figure 2 some tracking results of our methods in such a scenario are shown. Trajectories are computed within a predefined region of interest (ROI). Inside the larger (yellow) ROI new potential trajectories are started but only inside the smaller (green) ROI they are allowed to become valid trajectories). Each track gets a unique identifier assigned. Furthermore the calculated height of the persons in centimeters is shown. In the upper left corner the current person count is presented which is simply the sum of all valid tracks.

In Figure 2(a) the trajectories of two persons walking close to each other (piggybacking) are shown. Although the right person crouches just beside the taller left person the algorithm is still able to correctly detect and track the two persons. Figure 2(b) shows 3 persons piggybacking and the algorithm was able to correctly detect and track each individual person. In Figure 2(c) and 2(d) cases are shown where we tried to distract the system with a hat. The head has similar depth values like a human head. What can be seen is that a hat alone without depth values corresponding to human shoulders does not result in a wrong detection (Figure 2(c)). When the hat is carried at the height of a human head and close to a human's shoulder it produces a wrong detection and distracts the tracker by assigning it the track of a previous tracked head (Figure 2(d)). Figure 2(e) and 2(f) show two examples of humans with luggage and carrying a backpack which does not result in wrong detections or false tracks.

5. Conclusion and Future Work

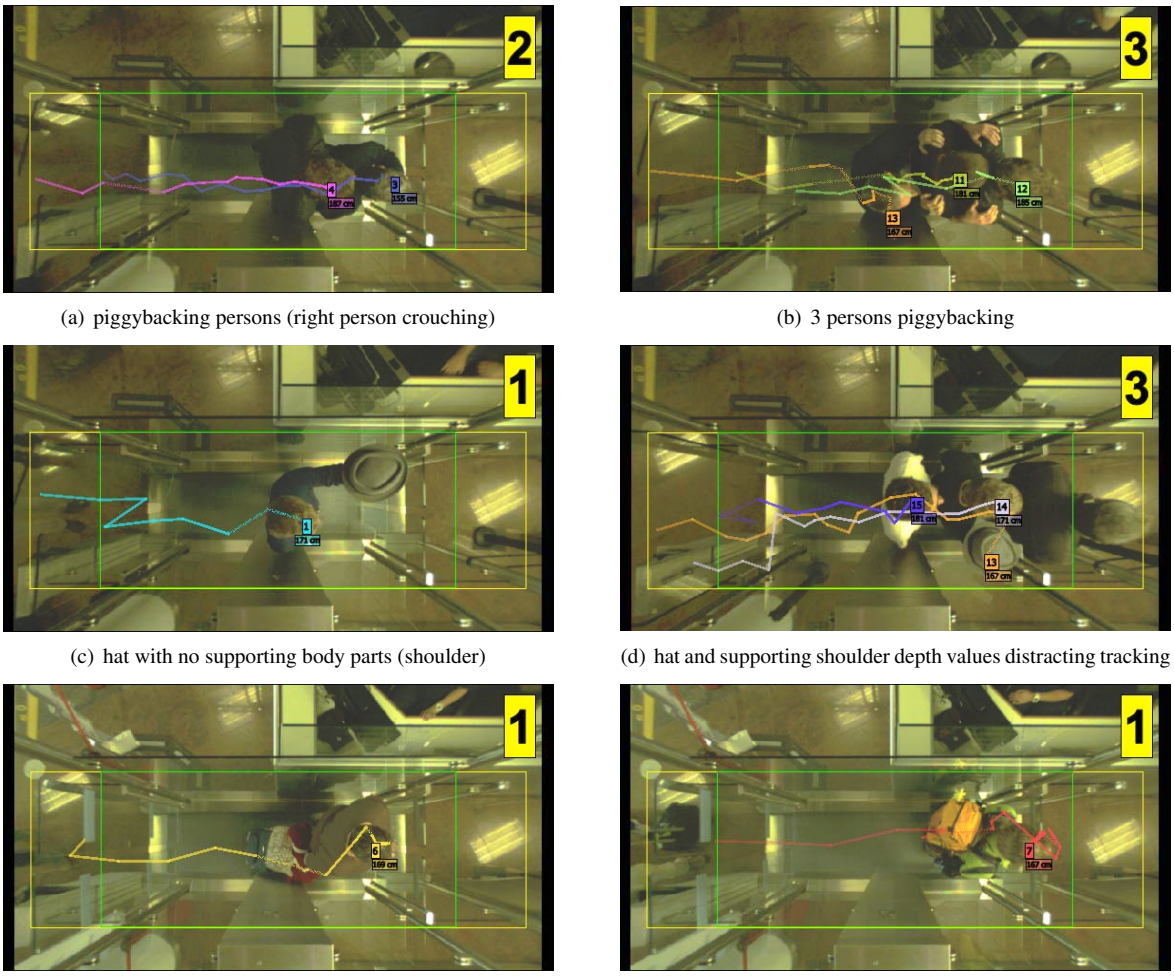
We presented a method to detect, track and count humans in top-view depth images. We described the algorithmic steps involved, especially our findings regarding adequate features and their impact on the classification performance. It was shown that we could successfully solve particular difficult scenarios with persons piggybacking and tailgating. Since our algorithm does not depend on background modeling it does not suffer from its shortcomings. This makes our approach very robust and suited for real-world scenarios. We gave an overview of the runtime of the algorithm.
We can think of several extensions and improvements to our algorithm. A combination of the presented feature descriptor with other features like e.g. HOG features on color data could further increase detection performance. A further enhancement of the computation performance of our algorithm could be achieved by utilizing several processor cores for algorithmic parts, e.g. the gradient climbing computation in the head localization step could be done in parallel for every seed point.

References

[1] D. Beymer. Person counting using stereo. In *Proceedings of the Workshop on Human Motion*, pages 127–133, 2000.

[2] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[3] D. Comaniciu, P. Meer, and S. Member. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619, 2002.



(a) piggybacking persons (right person crouching)

(b) 3 persons piggybacking

(c) hat with no supporting body parts (shoulder)

(d) hat and supporting shoulder depth values distracting tracking

(e) person with luggage not affecting detection quality

(f) person with backpack not affecting detection quality

Figure 2. Resulting trajectories of our method in a framework for tracking and counting humans in a gate system

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005.

[5] X. Huang, L. Li, and T. Sim. Stereo-based human head detection from crowd scenes. In *International Conference on Image Processing*, pages 1353–1356 Vol.2, 2004.

[6] R. Luo and Y. Guo. Real-time stereo tracking of multiple moving heads. In *Proceedings of the IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pages 55–59, 2001.

[7] S. Park and J. Aggarwal. Head segmentation and head orientation in 3d space for pose estimation of multiple people. In *Proceedings of the 4th IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 192–196, 2000.

[8] T. Pham. Non-maximum suppression using fewer than two comparisons per pixel. In *Advanced Concepts for Intelligent Vision Systems*, volume 6474 of *Lecture Notes in Computer Science*, pages 438–451. 2010.

[9] L. Spinello and K. Arras. People detection in rgb-d data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3838–3843, 2011.

[10] T. van Oosterhout, S. Bakkes, and B. J. A. Kröse. Head detection in stereo data for people counting and segmentation. In *Proceedings of the Sixth International Conference on Computer Vision Theory and Applications*, pages 620–625, 2011.

[11] P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, pages 511–518, 2001.

[12] S. Wu, S. Yu, and W. Chen. An attempt to pedestrian detection in depth images. In *Intelligent Visual Surveillance, 2011 Third Chinese Conference on*, pages 97–100, 2011.

[13] T. Yahiaoui, C. Meurie, L. Khoudour, and F. Cabestaing. A people counting system based on dense and close stereovision. In *Proceedings of the 3rd international conference on Image and Signal Processing*, pages 59–66, 2008.

[14] S. Yu, S. Wu, and L. Wang. Sltp: A fast descriptor for people detection in depth images. In *International Conference on Advanced Video and Signal-Based Surveillance*, pages 43–47, 2012.