

People-Flow Counting Using Depth Images for Embedded Processing

Guilherme S. Soares^(✉), Rubens C. Machado, and Roberto A. Lotufo

School of Electrical and Computer Engineering, University of Campinas,
Campinas, SP, Brazil

soares.g@gmail.com, rubens.campos.machado@gmail.com,

robertoalotuf@gmail.com

<http://www.feec.unicamp.br>

Abstract. This paper proposes a people-flow counting algorithm using top-view depth images for implementation on low-power, embedded processors. In the people detection stage the algorithm uses morphological connected filters to find head candidates, and in the tracking stage it uses Kalman filtering in order to obtain good predictions in frames where detection fails. A fast interpolation algorithm is also proposed, which estimates the values of pixels affected by noise and generates an image with a continuous domain. The experiments were done using a Kinect sensor and the processing was performed in real time on a Raspberry Pi 3. The dataset consisted of 4025 short video sequences of people entering and exiting indoor environments, obtained from three different installations. The algorithm proved to be adequate for an embedded application, reaching an accuracy of 98% for frame rates as low as 5.5 FPS.

Keywords: People counting · Depth images · Embedded processing

1 Introduction

The increasing availability of different types of sensors at progressively lower costs is allowing the improvement of many solutions involving automatic people detection, and it is also creating newer demands. Applications include surveillance, security, person-count estimation in public spaces and retail analysis. The types of sensors often used range from simple infrared beam cut detectors to more sophisticated sensors such as RGB and depth cameras.

Many techniques have been proposed for solving problems related to automatic people counting. Most of these techniques rely on conventional PCs to perform the processing. Examples include [9, 10] for single RGB cameras, [1] for stereo RGB cameras, and [2, 7] for depth cameras.

A few other studies included the particular requirement of performing low-power embedded processing. In [8], a depth camera was used to perform static counting of the number of people in a room, with the processing being made on a Raspberry Pi 2. In [4] a network of depth sensors was used, with sensor

connected to a Raspberry Pi 2, in order to detect and track people around a scene; the accuracy thus obtained was around 95%.

In this paper we propose an algorithm for automatic people-flow counting using top-view depth images. We also implement and test this algorithm on a Raspberry Pi 3, performing real time processing, using a Kinect depth sensor to capture the images.

2 Algorithm Overview

The proposed algorithm is composed of four stages: preprocessing, in which low-level image processing is performed in order to prepare the image for further analysis; people detection, which attempts to identify the coordinates of all the people present in a single frame; tracking, which attempts to establish a correspondence between the coordinates of people in consecutive frames and determine their trajectory; and counting, which analyzes each trajectory and determines whether the movement is an entrance, an exit, or neither. These steps are executed in sequence for each frame of the video. They are described in more detail further ahead.

2.1 Preprocessing

The preprocessing stage comprises four steps. The first step is a subsampling operation, resulting in an image of 320×240 pixels, in order to achieve faster processing; in the second step the 10-bit raw disparity values provided by the Kinect are mapped into an 8-bit linear image, so that the resulting values are proportional to the distance, in meters, from the sensor to the corresponding points in space; in the third step, the image is negated; and in the fourth step, pixels with unknown values are estimated using an interpolation algorithm.

Interpolation Algorithm. The depth image retrieved by the Kinect sensor typically presents some inherent noise, which leads to pixels with unknown depth values. It is mostly caused by interference of light sources, extremely dark surfaces, translucent objects, specular reflections and occlusions that lead to shadows in the infrared projection. Even though the depth values of pixels with noise cannot be determined, these pixels can be correctly identified by the sensor, which allows us to use methods that attempt to fill in those gaps.

The following method performs an interpolation of valid pixels by setting all invalid pixels to 0 and applying successive conditional dilations. The conditioning image is the mask of all invalid pixels at each iteration. The dilation is repeated until all the initially invalid pixels have a value assigned to them, as shown in Fig. 1. This process is different from the morphological reconstruction in that each pixel has its value changed only once, due to the conditioning image being updated at each iteration. Further details of conditional dilations and other morphological operations can be found in [5].

Morphological operations can have very time-consuming implementations. In this application, we achieve a fast execution by first calculating the distance transform, as in [3], on the invalid pixels, thus obtaining the distance between each invalid pixel and its closest valid one. Next, we sort all the invalid pixels by their distances, in ascending order, and then we sequentially apply the dilation to these pixels simply by assigning to each one the maximum value of its valid neighbors.

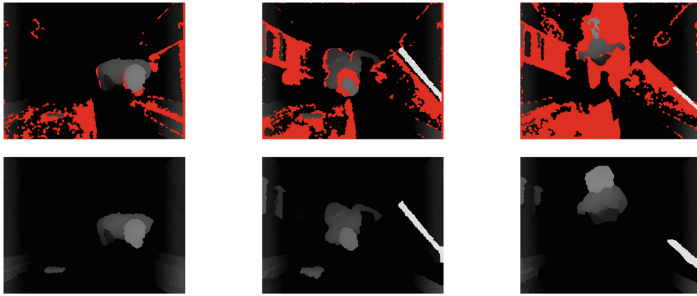


Fig. 1. Examples of the interpolation algorithm for images with different levels of noise. The areas in red represent the pixels with unknown values, as provided by the Kinect sensor. (Color figure online)

2.2 People Detection

This stage of the algorithm consists in attempting to identify the coordinates of all people present in a single image. This is achieved by a two-step process: first, all the regional maxima in the image are retrieved; next, these regional maxima are filtered out according to some parameters, so that only those that are more likely to correspond to people will remain.

According to [5], a regional maximum is a flat zone not adjacent to a flat zone with higher gray value. It is different from a local maximum in that regional maxima are a property of a region, and not of a single pixel. Due to the fine variations in the level of gray throughout the image, some filtering is necessary to retrieve relevant regional maxima, otherwise they would be too many and would correspond to very small regions. By using the operator h -maxima, described in [5], we obtain much more relevant regional maxima, that actually correspond to meaningful objects in the scene.

After obtaining all the regional maxima filtered with the h -maxima operator, the next step consists in selecting the ones that are more likely to correspond to the heads of the people in the scene. This is done by analyzing two criteria of each regional maximum: it must have a minimum area, which eliminates objects that are too small; and it must have a minimum aspect ratio, which eliminates objects that are too elongated. The centroids of the selected regions are then calculated as inputs in the tracking stage.

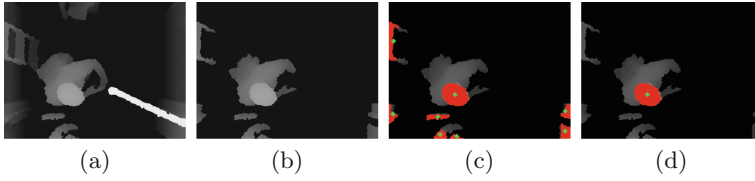


Fig. 2. Steps of the people detection stage: (a) input image; (b) thresholding and area open operations; (c) regional maxima after h -maxima operation; and (d) selected regional maxima.

In order to achieve better results, we also apply a thresholding and an area open operations before detecting the regional maxima. The thresholding helps eliminating objects that are too low or too high to match the height of a person's head, and the area open eliminates noise that could appear after the thresholding operation. Figure 2 summarizes the final pipeline of the people detection stage.

2.3 Tracking

The tracking stage is responsible for determining the motion of each person in the scene by establishing the correspondence between the detected centroids in two consecutive video frames. The approach used here only takes into account the spatial coordinates of the centroids; further improvements could also incorporate in the matching process other information about the associated regions, such as shape, area, etc.

Prediction and Matching. In order to achieve a better accuracy in matching the centroids, a Kalman filter [6] is used for each centroid to predict its position in the next frame. It also has the effect of correcting the measured position, thus reducing the measurement noise and producing smoother trajectories. Furthermore, for sequences where a person fails to be detected in some frames, the Kalman filter allows their position to be estimated in those frames, which is essential to keep their tracking.

The first step in the tracking stage is to perform the prediction, using the Kalman filter, of all the centroids that were present in the scene in the previous frame. After that, each predicted centroid is compared to all currently detected centroids, and their Euclidean distances are calculated. The result is an $m \times n$ matrix D with all the pairwise distances, where m is the number of predicted centroids from the previous frame and n is the number of detected centroids in the current frame. The elements of D are then analyzed one at a time, in ascending order of distance; for each element $d_{i,j}$, a match between centroids i and j is established if the following criteria are verified:

- distance $d_{i,j}$ is within a maximum distance $d_{\max}$;
- centroids i and j have not matched any centroids yet.

Once all the matches have been determined, the centroids from the current frame that have a correspondence in the previous one have their position corrected by the Kalman filter. Those that do not have a match are assumed to have entered the scene in the current frame, and a new Kalman filter is initialized for each one. Finally, the predicted centroids from the previous frame that do not have a match are propagated to the current frame, except for those having already reached a maximum number of consecutive propagations; those are assumed to have left the scene and are therefore discarded. Figure 3 illustrates the main steps of the tracking stage.

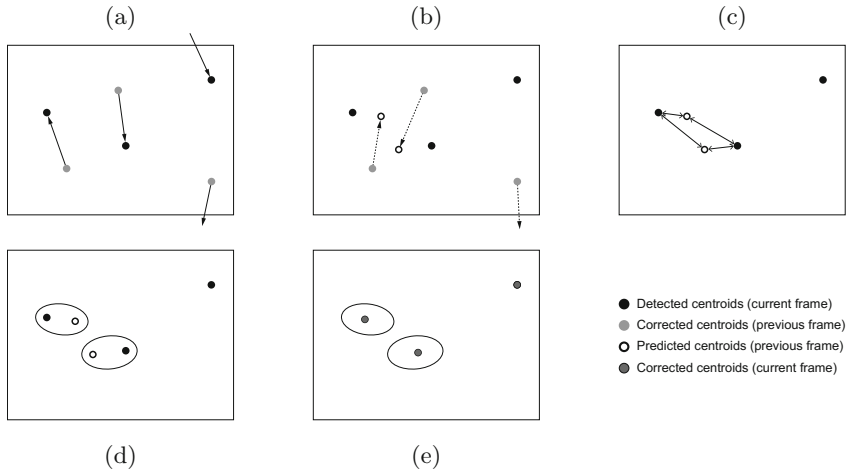


Fig. 3. Steps of the tracking stage: (a) true motion of centroids between two consecutive frames; (b) predictions made by the Kalman filter; (c) distances below the threshold $d_{\max}$; (d) matches between centroids; (e) corrections made by the Kalman filter.

Kalman Filtering. In the Kalman filter theory, the generic model for the evolution of the true state $\mathbf{x}$ of a system, at time k , is

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{G}_k \mathbf{u}_k + \mathbf{w}_k \quad (1)$$

where $\mathbf{F}_k$ is the transition model, $\mathbf{G}_k$ is the control-input model and $\mathbf{w}_k$ is the process noise. The observation $\mathbf{z}_k$ of the true state is given by

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \quad (2)$$

where $\mathbf{H}_k$ is the observation model and $\mathbf{v}_k$ is the observation noise.

In this particular application, the dynamic system model is quite simple: it assumes that the predicted position of a person depends on their current position and their current velocity, and that in the absence of an input (acceleration)

their predicted velocity will remain constant. The actual acceleration, which is unknown, is entirely modeled by the process noise $\mathbf{w}_k$. Thus, Eq. (1) becomes

$$\underbrace{\begin{bmatrix} x_k \\ y_k \\ z_k \\ \dot{x}_k \\ \dot{y}_k \\ \dot{z}_k \end{bmatrix}}_{\mathbf{x}_k} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{F}_k} \underbrace{\begin{bmatrix} x_{k-1} \\ y_{k-1} \\ z_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \\ \dot{z}_{k-1} \end{bmatrix}}_{\mathbf{x}_{k-1}} + \mathbf{w}_k \quad (3)$$

where Δt is the time interval between two consecutive frames.

As to the observation model, the only quantity we can measure is the position of the centroid. Therefore, Eq. (2) can be written as

$$\underbrace{\begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix}}_{\mathbf{z}_k} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{H}_k} \underbrace{\begin{bmatrix} x_k \\ y_k \\ z_k \\ \dot{x}_k \\ \dot{y}_k \\ \dot{z}_k \end{bmatrix}}_{\mathbf{x}_k} + \mathbf{v}_k \quad (4)$$

Loss of Frames. The capture frame rate of the Kinect sensor is 30 FPS. In practice, however, it is not possible to ensure that all captured frames are processed in a real-time application. Limited processing power, along with other hardware limitations, result in a slower, non-constant processing frame rate. Since the tracking is a time-dependent stage, it must take into account the actual time ΔT elapsed between two processed frames. In particular, the following aspects should be observed:

- The prediction phase of the Kalman filter should be repeated n times; n represents the *frame step* between two iterations of the algorithm, i.e., the number of frames captured since the last processed frame, and is given by

$$n = \frac{\Delta T}{\Delta t} \quad (5)$$

- As uncertainty increases with the number of consecutive predictions, the thresholding distance $d_{\max}$ should give more tolerance as n increases, and can be set as

$$d_{\max} = v_{\max} \Delta T \quad (6)$$

where $v_{\max}$ is the maximum speed assumed for a person.

3 Experiments

3.1 Dataset

Three different installations were used in the experiments, with different characteristics, as shown in Table 1 and Fig. 4. As the images were captured, they were processed and recorded, along with the outputs of the algorithm. In a posterior analysis the data were split into 4025 short video sequences, and each sequence was manually labeled with their corresponding number of entrances and exits.

Table 1. Characteristics of the three installations.

	Installation A	Installation B	Installation C
Sensor height (H)	2.90 m	3.80 m	4.00 m
Zenith angle (θ)	5°	15°	15°
Door type	Hinged	Hinged	Sliding
Door material	Glass	Glass	Glass
Room type	Hallway	Open room	Open room
Direct sunlight	Yes	Yes	No
Sunlight source	Skylight above	Front door	-

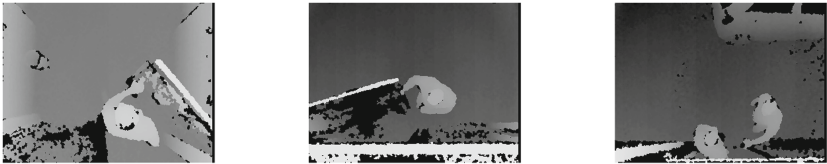


Fig. 4. Sample images of each installation.

3.2 Results

The processing was performed in real time by a Raspberry Pi 3, which has a 1.2 GHz quad-core ARM processor and 1 GB of RAM. The results of the experiments are shown in Table 2. The average frame rate, for the real-time execution a

Table 2. Results of the experiments for the three installations.

	Valid pixels	TP	FN	FP	Recall	Precision	Accuracy
Installation A	83.0%	964	25	14	0.97	0.99	0.96
Installation B	66.8%	807	147	10	0.85	0.99	0.84
Installation C	86.8%	1462	14	16	0.99	0.99	0.98

the Raspberry Pi 3, was around 5.5 FPS. As can be seen, the maximum accuracy of the system in a particular installation is closely related to the quality of the image acquisition, which is evidenced by the average percentage of valid pixels per frame.

4 Conclusions

Our experiments, conducted in real case scenarios, demonstrate the efficacy of using a depth sensor to perform high-accuracy people-flow counting with real-time, embedded processing. With an accuracy of up to 98%, our algorithm brings results comparable to other experiments of people-flow counting found in the literature, while being able to perform the entire computation on a low-power single-board computer.

References

1. Bernini, N., Bombini, L., Buzzoni, M., Cerri, P., Grisleri, P.: An embedded system for counting passengers in public transportation vehicles. In: 2014 IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA), pp. 1–6, September 2014
2. Bondi, E., Seidenari, L., Bagdanov, A.D., Del Bimbo, A.: Real-time people counting from depth imagery of crowded environments. In: 2014 11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp. 337–342. IEEE (2014)
3. Borgefors, G.: Distance transformations in digital images. *Comput. Vis. Graph. Image Process.* **34**(3), 344–371 (1986)
4. Burbano, A., Bouaziz, S., Vasiliu, M.: 3D-sensing distributed embedded system for people tracking and counting. In: 2015 International Conference on Computational Science and Computational Intelligence (CSCI), pp. 470–475, December 2015
5. Dougherty, E.R., Lotufo, R.A.: Hands-on Morphological Image Processing. SPIE Tutorial Texts in Optical Engineering, vol. TT59. SPIE Publications, Bellingham (2003)
6. Kalman, R.E.: A new approach to linear filtering and prediction problems. *J. Basic Eng.* **82**(1), 35–45 (1960)
7. Malawski, F.: Top-view people counting in public transportation using Kinect. *Chall. Mod. Technol.* **5**, 17–20 (2014)
8. Sgouropoulos, D., Spyrou, E., Siantikos, G., Giannakopoulos, T.: Counting and tracking people in a smart room: an IoT approach. In: 2015 10th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP), pp. 1–5, November 2015
9. Yang, D.B., Gonzlez-Baos, H.H., Guibas, L.J.: Counting people in crowds with a real-time network of simple image sensors. In: 2003 Proceedings of the Ninth IEEE International Conference on Computer Vision, pp. 122–129. IEEE (2003)
10. Zhang, E., Chen, F.: A fast and robust people counting method in video surveillance, pp. 339–343. IEEE, December 2007