Indian Journal of Science and Technology, *Vol 9(48), DOI: 10.17485/ijst/2016/v9i48/109311, December 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Field Seeding Algorithm for People Counting Using KINECT Depth Image

Berlian Akbar Yon Agusta[1], Pradit Mittrapiyanuruk[2] and Pakorn Kaewtrakulpong[3*]

[1]School of Electrical Engineering and Informatics, InstitutTeknologi Bandung, Indonesia; akbarberlian@gmail.com
[2]Department of Computer Science, Faculty of Science, Srinakharinwirot University, 114 Sukhumvit 23, Bangkok – 10110, Thailand; praditm@g.swu.ac.th
[3]Department of Control System and Instrumentation Engineering, King Mongkut's University of Technology, Thonburi 126 Pracha-utid Road, Bangmod, Tungkaru, Bangkok – 10140, Thailand; pakorn.kae@kmutt.ac.th

Abstract

In this work, we present a people counting algorithm using depth images acquired from a KINECT camera that is installed vertically, i.e., pointing toward the floor. Our proposed algorithm is referred to as Field seeding algorithm. The key idea is that first a set of local minimum values are detected from several spatially distributed seed locations. Then, the people-head blobs are detected from the binary images generated with regard to the threshold values derived from the local minimum values. The recall, accuracy and F-score of our algorithm are comparable to the current state-of-the-art people counting using KINECT, i.e. Water Filling. However, the main advantage over the previous method is that our algorithm operates deterministically, i.e., no any random number generating function is used.

Keywords: Depth Image, Head Detection, People Counting, Vertical Kinect

1. Introduction

People counting application is normally used for measuring a number of people passing in a specific area. This information can be used for several applications, e.g., people flow monitoring and analysis. In real life application, this system usually installed at the door of entrance area to count the number of people moving into and out of the area, for instance, in subway, stations, and department stores.

2. Related Work

In relevant literature, some methods[1–4] based on regression technique is proposed. These methods learn a mapping between extracted features and the number of people from the training images. Then, the mapping is used to estimate the number of people in an input image.

In[5–7], the blob tracking based methods is proposed. The blobs corresponding to people in an image are extracted by using background subtraction technique followed by binary image analysis on the difference image.

Methods using depth image are proposed in[8–10]. Using depth image for people counting problem can simplify the way to design algorithms in which several issues, e.g., occlusion, illumination variation, scene clutter, are easier to tackle.

Kinect sensors are used to provide the depth images for people counting algorithms in[10,11]. Kinect device was launched by Microsoft. It was originally used as a game control device. Based on the structure light technique, the Kinect provides two kinds of images: RGB image and depth image. That is, in each pixel of image, we obtain the depth information as well as the R-, G-, B-intensities of corresponding scene point.

In this paper, we propose a people counting algorithm by analyzing the depth images obtained from Kinect camera that is placed vertically so that the camera points toward the ground as shown in Figure 1. This camera configuration is similar to the previous work proposed in[10].

*Author for correspondence

However, our proposed algorithm, which is referred to as *Field Seeding* algorithm, operates deterministically. That is, our method does not use any random number generating function as the current state-of-the-art KINECT people counting system, i.e., the Water Filling algorithm proposed in[10]. Resorting to such random functions as in[10] makes the behavior of algorithm unpredictable.

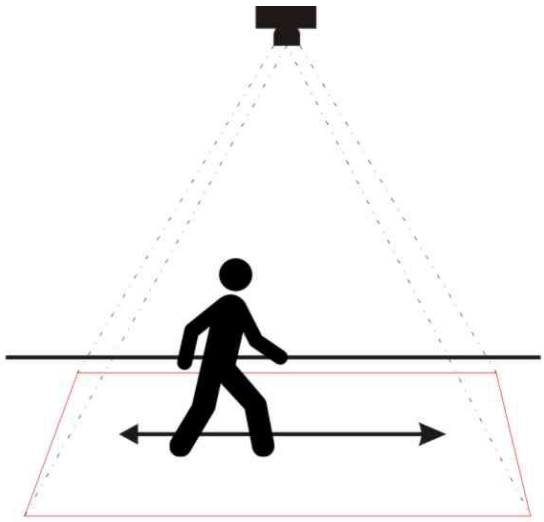


Figure 1. Kinect camera placement of "vertical Kinect".

The remainder of this paper is organized as follows. In Section 2, we present our proposed algorithm for people counting. Then we report the experimental result in Section 3. Finally, the conclusion is drawn in Section 4.

3. Our Proposed Algorithm

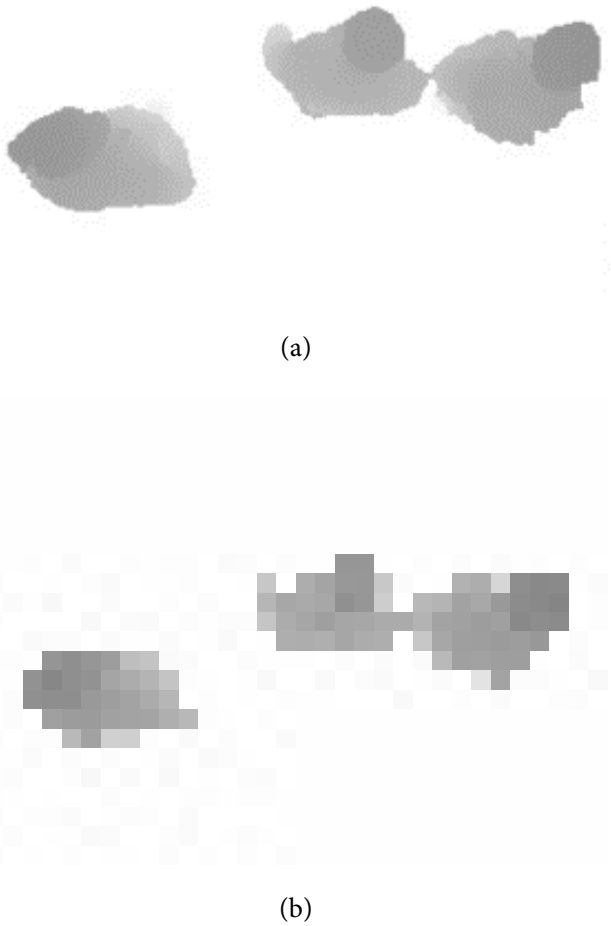
In the same spirit as[10], the problem of detecting people in a depth image can be formally stated as follows.

We denote a function f as the input depth image. And, $f(x,y)$ is the depth of pixel (x,y). Detecting people in depth image is equivalent to finding local minimum regions in f that satisfy

$$E_A(f(x,y)) \leq E_{N\setminus A}(f(x,y)) \quad (1)$$

where A is a local region and N is its neighborhood. $E_r(\cdot)$ is an operation that returns a value reflecting the total depth in the region r.

The key idea of our *Field Seeding* algorithm is to extract the blobs from the binary images generated from the input depth image with regard to different local minimum values in the neighborhood areas of spatially distributed seeding locations, i.e., field seeding.



(a)

(b)

Figure 2. (a) Input depth image and (b) pre-processed image (rescaled from $1/B^2$ of original size for better visualization).

Generally, the locations corresponding to these local minimum values could potentially be detected as people heads. Therefore, by considering only the pixels of depth image that their depth values are closed to a local minimum value, the resulting binary image could distinctively show people head blobs that can be easily extracted.

Our proposed algorithm can be presented as pseudo-code in Algorithm 1. The details are explained as follows.

We perform some pre-processing steps on the input depth image f and then obtain the output image g. These steps correspond to the function ***Preprocess*** in Line 1. First, we alleviate the effect of noise in the depth image by

applying the inpainting technique[12] to fill in the holes in depth data (i.e., missing data due to imperfect Kinect 3D reconstruction). Then, for the purpose of increasing the processing speed, we reduce the size of input depth image by subsampling every non-overlapped $B \times B$ block of input image into a pixel of output image. To specify a suitable value for B, we consider the criteria that the value of B must be smaller than the distance between two adjacent heads. This is to avoid the disruption that two heads could be combined into one in the output image. An example of the processing for this step is shown in Figure 2(a) and Figure 2(b).

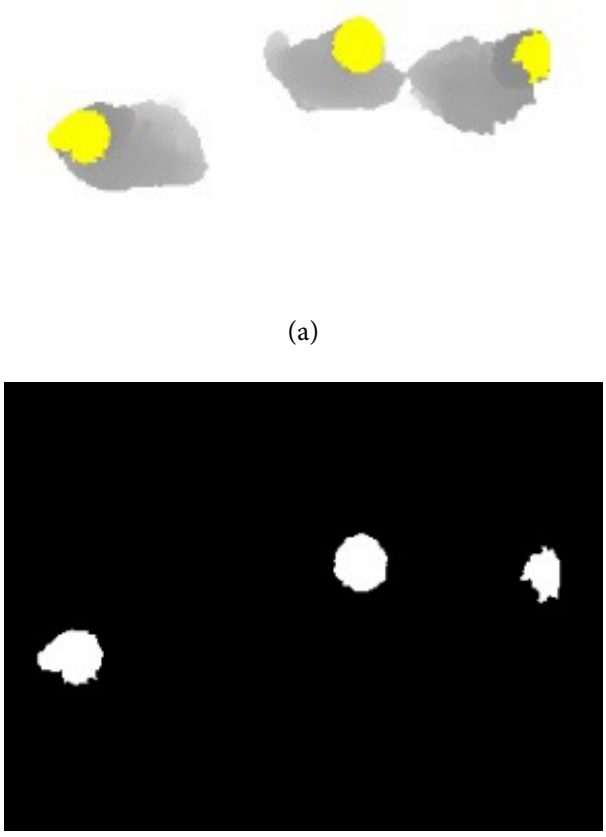
Algorithm 1: Our field seeding algorithm for people counting

```
Input: f = Depth image
Output: PeopleCnt = Number of people counts in f
        P = Set of people head blobs
 1  g ← Preprocess(f)
 2  Establish L non-random seed locations
    {(xk,yk); k=1, 2,..., L}
 3  P = ϕ
 4  for k=1 to L
    /* Find a local min in the neighborhood area
       NxN around (xk,yk) */
 5    tk ← FindLocalMin(g(xk±N/2,yk±N/2))
 6    bk ← Threshold(g,tk±t)
 7    {Ci} ← FindBlob(bk)
 8    for i=1 to |Ci|
 9      if IsHeadBlob(Ci) == true
10        P ← P ∪ Ci
11      end if
12    end for
13  end for
14  PeopleCnt = | P |
```

Next, as expressed in Line 3, we establish a set of L non-random seed locations $\{(x_k,y_k);\ k=1, 2,\ldots, L\}$ in the pre-processed image g. Specifically, for the depth images of size 320×240 used in our experiment, we set the value of B to 10 and we take every 4 pixels in the x and y directions as the seed locations.

For each seed location (x_k,y_k), as expressed by the for loop in Lines 4 to 13, we extract foreground regions corresponding to people heads from the binarized depth image. The binary image is generated by thresholding with regard to the local minimum value determined in the local region around (x_k,y_k).

That is, we determine the local minimum inside the neighborhood area of size $N \times N$ around (x_k,y_k) of the pre-processed depth image g. This step corresponds to the function ***FindLocalMin***in Line 5 in which we obtain the minimum value t_k. Note that, for the aforementioned configurations of values in our experiment, we set the value of N to 4.



(a)

(b)

Figure 3. Final result of detected people heads, (a) blob masks and (b) blob masks overlaid on the input depth image.

Then, we generate the binary image b_k from the pre-processed depth image g where the pixel value of $b_k$is 1 if the corresponding depth value in g is in the range t_k-t to t_k+t (the function ***Threshold*** in Line 6).

Subsequently, we extract a set of blobs, $\{C_i\}$, from the binary image b_k by the function ***FindBlob***. We check whether each blob, i.e. C_i, really corresponds to a person head by verifying region properties of the blob (the function ***IsHeadBlob*** in Line 9). Currently two properties are used. That is, first we verify whether the radius of the minimum enclosing circle of the blob is in a specified range (e.g., r_{min} to r_{max}). Second, we verify whether the ratio between the blob area and the area of minimum

enclosing circle is larger than a threshold, e.g. 50%. If both criterion are satisfied for each blob, (expressed in Line 10) we include the blob to the final result, i.e. the set *P*.

At the end of processing of all seed locations in Line 14, the number of people detected in the input depth image is equal to the number of blobs in the set *P*. An example of the final result can be seen in Figure 3.

4. Experiment

To compare our method to the previous method[10], we perform the experiment on the test data provided by the authors of[10] in which it consists of two datasets taken at two different scenes. The first dataset contains 2384 images with 4541 heads and the second dataset contains 1500 images with 1553 heads. The image size is 320x240 pixels.

As pre-processing, the adaptive background subtraction technique[13] is used to remove the effect of background so that the depth information of extracted foreground regions is further used in the detection.

The parameters setting for our algorithms are as follows. The value of *B* is set to 10 and we take every 4 pixels in the x and y directions as the seed locations. The value of *N* is set to 4. The value of *t* is set to 3. For the function ***IsHeadBlob***, the values of r_{min} and r_{max} are set to 11 and 20, respectively. Also, the threshold of the ratio between the blob area and the area of minimum enclosing circle is set to 50%.

We use three indicators to identify the performance, i.e., Recall, Accuracy, and F-Score that can be shown by the following equations:

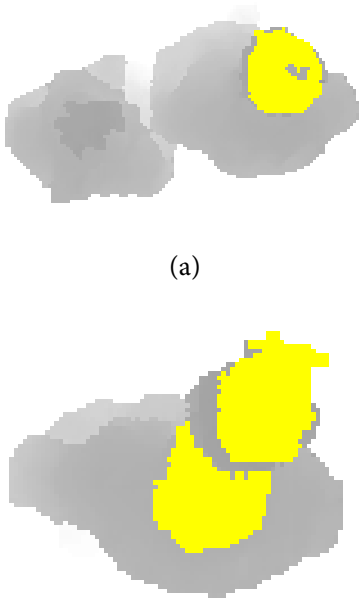
$$\text{Recall} = \frac{\text{Number of correctly detected heads}}{\text{Number of actual heads in dataset}} \quad (2)$$

$$\text{Accuracy} = \frac{\text{Number of correctly detected heads}}{\text{Number of all detected heads}} \quad (3)$$

$$\text{F-Score} = 2\frac{\text{Accuracy*Recall}}{\text{Accuracy+Recall}} \quad (4)$$

The results of our algorithm on the datasets can be shown in Table 1. From the result, we have found that our algorithm performs very well in which the recall rates are 99.41% for the first dataset and 99.29% for the second dataset. And, the accuracies are 95.53% and 92.56% for the first and second datasets, respectively. The F-Scores are 97.4% and 95.8% for both datasets.

A failure case we found is according to the result of preprocessing step. As shown Figure 4(a), the left person's head cannot be detected due to the noise in depth image. As we can see in the figure, the shape of the head in the left is not quite round. This is due to some improper values that are filled by our preprocessing step.



(a)

(b)

Figure 4. Failure cases, (a) result of detection error due to improper working on preprocessing step (b) Error due to that a person's back is detected as another head.

Another failure case is shown in Figure 4(b), the algorithm incorrectly detect the person's back as a head. We call this problem as a "*head criteria problem*". This problem is according to the criterion used in the function ***IsHeadBlob***. Currently we use two simple conditions. In

Table 1. The result of our algorithm

Dataset	Number of heads in dataset	Number of detected heads	Number of correctly detected heads	Accuracy	Recall	F-Score
1	4541	4725	4514	0.9533	0.9941	0.9743
2	1553	1666	1542	0.9256	0.9929	0.9581

future work, we plan to improve the criterion so that it can handle such error.

5. Conclusion

We present a people counting algorithm using depth images acquired from a KINECT camera that is installed vertically, i.e., pointing toward the floor. Then, the problem of detecting people heads is recast as finding the local minima in each depth image.

Our algorithm is referred to as Field seeing algorithm. A set of local minimum values are detected from several spatially distributed seed locations. The people-head blobs are detected from the binary images generated with regard to the threshold values derived from the local minimum values. Our algorithm gives very good result in which it provides 99% in recall, 92% in accuracy and 95% in F-Score. The main advantage over the current state-of-the-art people counting using KINECT[10] is that our algorithm operates deterministically, i.e., no any random number generating function is used.

6. Acknowledgements

This work is supported by the Higher Education Research Promotion and National Research University Project of Thailand, OHEC.

7. References

1. Cho S, Chow T, Leung C. A neural-based crowd estimation by hybrid global learning algorithm. Institute of Electrical and Electronics Engineers (IEEE) Transactions on Systems, Man, and Cybernetics. 1999; 29(4):535–41.
2. Dong L, Parameswaran V, Ramesh V, Zoghlami I. Fast crowd segmentation using shape indexing. Institute of Electrical and Electronics Engineers (IEEE) 11th International Conference on Computer Vision; 2007 Oct. p. 1–8.
3. Kong D, Gray D, Tao H. Counting pedestrians in crowds using viewpoint invariant training. In Proceedings of the British Machine Vision Conference (BMVC), Oxford, UK; 2005 Sep.
4. Marana A, Costa L, Lotufo R, Velastin S. On the efficacy of texture analysis for crowd monitoring. International Symposium on Computer Graphics, Image Processing, and Vision (SICGRAPI); 1998 Oct. p. 354–61.
5. Velipasalar S, Tian YL, Hampapur A. Automatic counting of interacting people by using a single uncalibrated camera. Institute of Electrical and Electronics Engineers (IEEE) International Conference on Multimedia and Expo; 2006 Jul. p. 1265–8.
6. Kong D, Gray D, Tao H. A viewpoint invariant approach for crowd counting. The 18th International conference on Pattern Recognition. 2006 Aug 20–24; 4:630–3.
7. Chen TH, Chen TY, Chen ZX. An intelligent people-flow counting method for passing through a gate. Institute of Electrical and Electronics Engineers (IEEE) Conference on Robotics, Automation and Mechatronics; 2006 Jun 1–3. p. 1–6.
8. Terada K, Yoshida D, Oe S, Yamaguchi J. A counting method of the number of passing people using a stereo camera. The 25th Annual Conference of the Institute of Electrical and Electronics Engineers (IEEE). 1999 Nov 29 – Dec 3; 3:1318–23.
9. Beymer D. Person counting using stereo. In the Proceedings of Institute of Electrical and Electronics Engineers (IEEE) Workshop on Human Motion; 2000. p. 127–33.
10. Zhang X, Yan J, Feng S, Lei Z, Yi D, Li S. Water filling: unsupervised people counting via Kinect sensor. Institute of Electrical and Electronics Engineers (IEEE) Ninth International Conference on Advanced Video and Signal-Based Surveillance (AVSS); 2012. p. 215–20.
11. Hsieh CT, Wang HC, Wu YK, Chang LC, Kuo TK. A Kinect based people flow counting system. International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS); 2012 Nov 4–7. p. 146–50.
12. Alexandru T. An image in painting technique based on the fast marching method. Journal of graphics tools. 2004; 9(1):23–34.
13. Kaewtrakulpong P, Bowden R. A real time adaptive visual surveillance system for tracking low-resolution colour targets in dynamically changing scenes. Image and Vision Computing. 2003 Sep; 21(10):913–29.