An effective graph and depth layer based RGB-D image foreground object extraction method

Zhiguang Xiao¹, Hui Chen¹ (✉), Changhe Tu², and Reinhard Klette³

© The Author(s) 2017. This article is published with open access at Springerlink.com

We consider the extraction of accurate silhouettes of foreground objects in combined color image and depth map data. This is of relevance for applications such as altering the contents of a scene, or changing the depths of contents for display purposes in 3DTV, object detection, or scene understanding. To identify foreground objects and their silhouettes in a scene, it is necessary to segment the image in order to distinguish *foreground regions* from the rest of the image, the *background*. In general, image data properties such as noise, color similarity, or lightness make it difficult to obtain satisfactory segmentation results. Depth provides an additional category of properties.

1 Proposed method

Our approach includes four steps: see Fig. 1. Firstly, graphs are built independently from color and depth information such that each node represents a pixel, and an edge $e_{p,q} \in E$ connects nodes p and q . We transform the color space into *CIELUV* space to measure differences between pixels, and use as a region merging predicate: two regions are merged if and only if they are clustered in both color and depth graphs, providing more accurate over-segmentation results. Secondly, depth maps

are partitioned into layers using a multi-threshold method. In this step, objects belonging to different depth ranges are segmented into different layers. Thirdly, seed points are specified manually to locate the foreground objects, and to decide which depth layer they belong to. Finally, we merge the over-segmented scene according to cues obtained in the previous three steps, to extract foreground objects with their accurate silhouettes from both color and depth scenes.

1.1 Improved graph-based algorithm with depth

Although there have been related studies over the past 20 years, image segmentation is still a challenging task. To obtain foreground objects, the first step of our approach is to obtain an over-segmented scene. We improve upon the graph-based approach of Ref. [1] in the following two ways:

Selection of color space. The first improvement concerns the color space used. *RGB* color space is often used because of its compatibility with additive color reproduction systems. In our approach, dissimilarity between pixels is measured by edge weights, which is calculated using Euclidean distance in *CIELUV* color space. Color differences, measured by Euclidean distances in *RGB* color space are not proportional to human visual perception; *CIELUV* color space is considered to be perceptually more uniform than other color spaces.

Fusion of color and depth. The second important aspect is that we combine color and depth information to provide more over-segmented results. In Ref. [1], the merging predicate is defined for regions R_1 and R_2 as

$$M(R_1, R_2) = \begin{cases} 1, & \text{if } d_b(R_1, R_2) < d_i(R_1, R_2) \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

1 School of Information Science and Engineering, Shandong University, Jinan 250100, China. E-mail: Z. Xiao, xiaozhg@live.com; H. Chen, huichen@sdu.edu.cn (✉).

2 School of Computer Science and Technology, Shandong University, Jinan 250100, China. E-mail: chtu@sdu.edu.cn.

3 School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Auckland 1142, New Zealand. E-mail: rklette@aut.ac.nz.

Manuscript received: 2017-02-17; accepted: 2017-07-12

Fig. 1 (a) Input color scene, (b) input depth map, (c) over-segmentation result, (d) selection of seed points (red line), (e) selected depth layer, and (f, g) extracted foreground object in color and depth.

where the minimum *internal difference* d_i is defined by

$$d_i(R_1, R_2) = \min\{w(R_1) + \tau(R_1), w(R_2) + \tau(R_2)\} \quad (2)$$

Here, d_b and w are the *between-region difference* and the *within-region maximum weight*, respectively, $\tau(R)$ is a threshold function which is based on the area of region R , and d_b and w are defined as follows:

$$d_b(R_1, R_2) = \min_{p \in R_1, q \in R_2, e_{p,q} \in E} \{\omega(e_{p,q})\} \quad (3)$$

$$w(R) = \max_{e \in E_R} \{\omega(e)\} \quad (4)$$

where edge weight $\omega(e)$ is a measure of dissimilarity between two pixels connected by edge e . An edge $e \in E_R$ connects two pixels in region R .

Exclusive use of color information is very likely to lead to under-segmentation, and this needs to be avoided. Conversely, depth information may provide additional clues for providing more accurate silhouettes of objects. Thus, we build a separate graph based on depth information. During the segmentation process, two regions are clustered if and only if they are allowed to cluster both in the color image graph and the depth map graph.

1.2 Seed point specification

Seed points are used to locate the foreground objects in both color image and depth map. Our approach allows a user to specify an individual object to be extracted as a foreground object by roughly drawing a stroke on the object. We sample points on the trajectory of the stroke as seed points.

Typically our approach can extract the specified object by indicating seed points in this way only once, but in some cases, to obtain a satisfactory result, repeated interaction might be needed. Therefore we define two kinds of seed points, those inside and outside an intended object, which we call *positive* and *negative seed points* (PSPs and NSPs), respectively.

Regions containing positive seed points are called *positive seed regions*. When we unify an over-segmented color image, we remove regions which

contain negative seed points (*negative seed regions*) to break continuity of regions, which are connected under constraints defined by depth layers; we maintain positive seed point regions as well as regions which are connected to them. Therefore, for each extraction task, a user may draw one or more strokes inside the foreground object for extraction, and pixels from the stroke are used as PSPs. Next, our approach provides an extraction result. If the result contains regions which should not be merged, the user may draw an NSP stroke in the joined regions to separate them, like using a knife to cut off redundant parts.

1.3 Depth layering

Given a depth map of a 3D scene, the purpose of depth layering is to segment the continuous depth map into several depth layers. We assume in this paper that depth values for a single indoor object are only distributed within a small range. This assumption may not be true in general but appears to be acceptable in our application.

We partition the depth map into depth layers in the form of binary images. A depth layer contains pixels in a range of depth values, and we consider these pixels as the foreground region (white) of the chosen depth layer. One depth layer is used to extract one foreground object. Therefore, the specified foreground object for extraction should lie inside the foreground region of the selected depth layer, as our approach merges an over-segmented scene based on the selected depth layer. If depth values of one object are not in a small range then the depth value interval of this object is out of the considered range of one depth layer, and have an integral object which is divided into more than one depth layer. In such a case, our approach is unable to select a proper depth layer to extract the integral object.

Inpainting for depth map. Before depth layering, we do some preprocessing of the depth map, called *inpainting*, to reduce artefacts caused by the capturing procedure.

Time-of-flight cameras or structured lighting methods (as used by Kinect) cannot obtain depth information in over or under-exposed areas, and are often inaccurate at silhouettes of objects. This results in an incomplete depth map that has inaccurate depth for some pixels, and no depth at all for others. Estimating the depth for these missing regions is an ill-posed problem since there is very little information to use. Recovering the true depth would only be possible with very detailed prior knowledge of the scene (or by using an improved depth sensor, such as replacing one stereo matcher by another one).

We use an intensive bilateral filter, proposed in Ref. [2], to repair the depth map. The depth of each pixel with missing depth information is determined by searching the $k \times k$ neighboring pixels for ones with similar color in the color image, and with a non-zero depth value. The search range varies until a fixed number of satisfactory neighboring pixels is reached. After completing all depth values of such pixels, a median filter is applied to the depth map, to smooth outliers in the depth map.

Segmentation of depth map. After inpainting in the depth map, next we segment the depth map into different layers such that each layer contains pixels in a range of depth values. The problem here is to decide how many layers should be used. If too few segmented layers are used, many over-segmented regions produced in the last step are probably in the same layer. Contrariwise, if the segmented layers are too many, a single over-segmented region is probably spread across more than one layer. Either case will make it difficult to distinguish whether a region belongs to a foreground object or not.

Our goal is that, in this step, those regions which overlap according to the seed points specified by the users, should be contained in the same layer. This agrees with the assumption that the user will usually specify most of the regions that belong to a foreground object. With this constraint, we segment the depth map into layers using an extended multi-threshold algorithm as proposed in Ref. [3]. Equation (5) outlines how to segment a depth map into a given number n of layers:

$$L_m(i, j) = \begin{cases} 1, & \text{if } T_m < D(i, j) < T_{m+1} \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where $D(i, j)$ is the depth value at pixel (i, j) , and T_m , for $0 \leq m \leq n - 1$, is the m th threshold

computed by an extended Otsu's multi-threshold method.

We propose a method to find a proper depth layer automatically for a foreground object in given depth map. First, we initialise n as being the maximum number of layers that sufficient for any 3D scene considered. Thus, for any given depth map, the proper number of layers should be in the range of $2-n$. Then we split the depth map repeatedly from 2 to n layers, in an ordered way, and obtain a series of segmented layers each represented by one binary image. Thus, for any given depth map, an optimal layer for a specified object should be in this set of segmented layers.

We define the pixels with value 1 in one binary image as being the foreground pixels; they comprise the foreground region of this layer. A layer is defined to be a *valid layer* if and only if all the positive seed points are in the foreground region in its corresponding binary image.

We sort all valid layers according to the total number of foreground pixels in the binary image. Our experimental results indicate that choosing the middle valid layer from the sequence defines a good choice, allowing the proper depth layer of the specified foreground object.

1.4 Merging over-segmented color regions

The improved graph-based algorithm provides a result that consists of homogeneous segments, while the purpose of our algorithm is to extract semantically meaningful objects from the background. Hence, over-segmented regions need to be merged. The basic idea of region merging combines similarity in depth and dissimilarity in color between adjacent regions.

Actually, one object often contains a variety of colors while it connects to the background region at the same depth. Therefore, we propose to group the regions on the basis of regional continuity which is established under the constraint of depth layers. Our regional continuity function is defined as follows:

$$C(k) = \begin{cases} 1, & \text{if } A_d(k) > T_A A_c(k) \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

where $A_d(k)$ is the area of overlap of the foreground of the selected depth layer with region k , $A_c(k)$ is the total area of region k , and T_A is an adjustable coefficient.

Based on this criterion, the region merging step starts with region labeling, to distinguish and count the area of each region. Firstly, each region is relabeled (approximately) for initialization. Secondly, for each pixel p , we find those pixels among its 8-adjacent neighbors which belong to the same region as p . We then update p by assigning the minimum label among those of the detected 8-adjacent neighbors and p itself. We repeat this procedure until no update occurs. After that, each region has a unique label, and the area of each region, as well as the area of the region overlapping with the foreground region of the selected depth layer, can be determined by counting. Next, regional continuity is constructed on the basis of Eq. (6). We modify the regional continuity to remove mis-connected regions: negative seed regions and regions that are connected to positive seed regions via negative seed regions, should be disconnected from positive seed regions. Finally, semantically meaningful object results are obtained by merging positive seed regions and regions connected to them.

2 Experimental evaluation

2.1 Qualitative analysis

Our approach was evaluated mainly on a large-scale hierarchical multi-view RGB-D object dataset

collected using a Kinect device. A recently published dataset, the RGB-D Scenes Dataset v2 (RGB-D v2), includes 14 scenes which cover common indoor environments. Depth maps for this dataset were recovered by the intensive bilateral filter mentioned in Section 2.3 before the depth-layering step. The MSR 3D Video Dataset (MSR 3D), and more complex RGB-D images used by Zeng et al. [4], were also employed to test our approach.

Objects and their silhouettes extracted by our approach are shown in Fig. 2. Although Kinect devices provide depth maps with large holes and significant noise, a well restored depth map and the segmented results demonstrate the robustness of our algorithm to noise in the depth images. From our results we conclude that our approach is able (for the test data used) to extract foreground objects from the different background scenes.

2.2 Quantitative analysis

Metrics including precision, recall, and F -measure (see Eqs. (7)–(9)) were also computed and interpreted to analyze our results quantitatively:

$$P = \frac{T_p}{T_p + F_p} \quad (7)$$

$$R = \frac{T_p}{T_p + F_n} \quad (8)$$

$$F_\beta = \frac{R(\beta^2 + 1)P}{\beta^2 P + R} \quad (9)$$

Fig. 2 Extraction results for scenes from different datasets. (A, B) Extracted silhouettes in color and depth images. (C, D) Extracted foreground objects in color and depth images.

Fig. 3 Quantitative analysis of different methods. MW: magic wand, GC: grab-cut [6], BL: graph-based on color information only (i.e., baseline method), FM: the fast mode of Ref. [5] with depth layer, MS: mean-shift color-depth with depth layer, Our: our approach. The horizontal axis represents different datasets.

where T_p is the number of correctly detected foreground object pixels, F_p represents non-foreground object pixels detected as foreground object pixels, and F_n means undetected foreground object pixels.

We extract ground truth manually to evaluate the results, and set $\beta = 1$ to calculate the F -measure as we consider that the recall rate is as important as precision. Performance measures were computed for different datasets to evaluate the approach's effectiveness on those different datasets. See Fig. 3 for quantitative analysis results for our approach (yellow).

2.3 Comparison with other methods

For a comparative evaluation of our approach, we also tested five other methods designed for extracting objects from scenes in datasets as used above. They are the magic wand in PhotoShop, grab-cut, the original graph-based algorithm in Ref. [1] with depth layers, a multistage, hierarchical graph-based approach [5] with depth layers, and an improved mean-shift algorithm with depth layers. See Fig. 4 for comparative results.

Qualitative results. Compared to magic wand, shown in Fig. 4(A), our approach (Fig. 4(F)) is able to reduce the amount of user interaction considerably, only with a single initialisation needed to complete an extraction task.

Grab-cut [6], in Fig. 4(B), is excellent in terms of simplicity of the user input, but for colorful scenes, the extraction process is difficult and more interactions are needed to indicate the foreground and the background. Moreover, the results lack discrimination in the presence of color similarity.

The above methods only use color information to extract foreground objects. For a further illustration of the performance of our approach, extraction results provided by methods in which color and depth are both applied are also compared with our approach. First, we take the original graph-based algorithm [1] with depth layers as a *baseline method* in our experiments: see Fig. 4(C). The graph-based algorithm generates over-segmented results. Then, regions are merged based on depth layer constraints and seed points. Comparing results shows the effectiveness of our improved graph-based method.

We also compare with results obtained by using an algorithm published in Ref. [5] which combines depth, color, and temporal information, and uses a multistage, hierarchical graph-based approach for segmenting 3D RGB-D point clouds. Because the scenes in our applications are static, we are able to use the fast mode (i.e., removing temporal information) of the algorithm of Ref. [5] for providing over-segmented results. The 3D point cloud data, as generated by the color scene and depth map, are used as input for this method. The foreground objects are extracted based on the previous result, seed points, and the depth layers. See Fig. 4(D) for results of this method following Ref. [5].

An improved mean-shift algorithm with depth layers, shown in Fig. 4(E), is another candidate used for testing. Depth information is first added to amend the mean-shift vector to over-segment the color scene. The over-segmented results are merged based on the seed points and depth layers.

Quantitative results. Figure 3 presents the precision rate, recall rate, and F -measure for the above methods on three different datasets. One of the merging constraints of our approach is based on the depth layer, and as the edges of objects in the depth map are not so accurate (usually a little outside the objects compared to the ground truth), in the extraction results, our approach may merge some pixels that do not belong to the ground truth. Some methods provide higher precision because their

Fig. 4 Foreground objects and silhouettes extracted by different methods in both color and depth. (A) Magic wand, (B) grab-cut [6], (C) graph-based on color information only (i.e., baseline method), (D) the fast mode of Ref. [5] with depth layers, (E) mean-shift color-depth with depth layers, and (F) our approach. (a) Interactions, (b, c) extraction results in color and depth.

extraction results are not integral and are almost contained within the ground truth. Thus, the precision rate of our approach is lower than that of some other methods. However, our approach offers more integral extraction results, which makes the recall rate higher than that of the others. The F -measure with $\beta = 1$ demonstrates that our approach performs better.

Amount of interaction. Figure 4 shows the interaction need by each method for each scene. For the magic wand, the red spots show the seed points specified by the users. The sizes and locations of the red spots should be chosen according to different foregrounds.

In grab-cut, a rectangular box is drawn around the foreground object. Red lines are seed points in the foreground, and blue lines are seed points in the background. When applying the grab-cut method to colorful scenes, for example, the scenes used by Zeng et al., more iterations and seed points are needed. We do not show all of the iterations of the grab-cut method on a scene used by Zeng et al. in Fig. 4; it is difficult to follow them visually. Seed points for the other four methods are specified by roughly drawing a stroke on the foreground. Red lines represent the seed points for the foreground, and blue lines represent the background.

There is no limitation on seed points in our method; we usually draw a stroke around the center of the specified foreground object, but this is not necessary. If the automatically selected depth layer is appropriate for extracting foreground objects, then no further seed points are needed. If not, then more positive seed points are required to specify other positions to be extracted as parts of foreground objects. Positions can be located according to the previously selected depth layer; therefore a user can coarsely add positive seed points around the located positions to obtain a proper depth layer. The user is able to obtain expected results by applying positive and negative seed points flexibly.

The extracted results of our approach remain fairly robust: the integrity of the objects is mostly retained while silhouettes are better preserved. Our approach outperforms in general the other approaches regarding the quality of results, with a reduced need for interaction.

Acknowledgements

The authors thank the editors and reviewers for their insightful comments. This work is supported by Key Project No. 61332015 of the National Natural Science Foundation of China, and Project

Nos. ZR2013FM302 and ZR2017MF057 of the Natural Science Found of Shandong.

References

- [1] Felzenszwalb, P. F.; Huttenlocher, D. P. Efficient graph-based image segmentation. *International Journal of Computer Vision* Vol. 59, No. 2, 167–181, 2004.
- [2] Li, Y.; Feng, J.; Zhang, H.; Li, C. New algorithm of depth hole filling based on intensive bilateral filter. *Industrial Control Computer* Vol. 26, No. 11, 105–106, 2013.
- [3] Otsu, N. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics* Vol. SMC-9, No. 1, 62–66, 2007.
- [4] Zeng, Q.; Chen, W.; Wang, H.; Tu, C.; Cohen-Or, D.; Lischinski, D.; Chen, B. Hallucinating stereoscopy from a single image. *Computer Graphics Forum* Vol. 34, No. 2, 1–12, 2015.
- [5] Hickson, S.; Birchfield, S.; Essa, I.; Christensen, H. Efficient hierarchical graph-based segmentation of RGBD videos. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 344–351, 2014.
- [6] Rother, C.; Kolmogorov, V.; Blake, A. “Grabcut”: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics* Vol. 23, No. 3, 309–314, 2004.

Zhiguang Xiao is a postgraduate at the School of Information Science and Engineering, Shandong University. He received his B.E. degree in electronics and information engineering from the College of Electronics and Information Engineering, Sichuan University. His research interests are

in graph algorithms, computer stereo vision, and image segmentation.

Hui Chen is a professor at the School of Information Science and Engineering, Shandong University. She received her Ph.D. degree in computer science from the University of Hong Kong, her bachelor and master degrees in electronics engineering from Shandong University. Her research interests

include computer vision, 3D morphing, and virtual reality etc.

Changhe Tu currently is a professor and the associate dean at the School of Computer Science and Technology, Shandong University. He obtained his bachelor, master, and Ph.D. degrees all from Shandong University. His research interests include geometric modelling and processing, computational geometry, data-driven visual computing, etc. He published papers on SIGGRAPH, Eurographics, ACM TOG, IEEE TVCG, CAGD, etc.

Reinhard Klette is a Fellow of the Royal Society of New Zealand and a professor at Auckland University of Technology. He was on the editorial board of the *International Journal of Computer Vision* (2007–2014), the founding Editor-in-Chief of the *Journal of Control Engineering and Technology* (2011–2013), and an Associate Editor of *IEEE PAMI* (2001–2008). He (co-)authored more than 300 publications in peer-reviewed journals or conferences, and books on computer vision, image processing, geometric algorithms, and panoramic imaging. He presented more than 20 keynotes at international conferences. Springer London published his book entitled *Concise Computer Vision* in 2014.

Open Access The articles published in this journal are distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

Other papers from this open access journal are available free of charge from <http://www.springer.com/journal/41095>. To submit a manuscript, please go to <https://www.editorialmanager.com/cvmj>.