Automatic Object Segmentation on RGB-D Data using Surface Normals and Region Similarity

Hamdi Yalin Yalic and Ahmet Burak Can
Computer Engineering Department, Hacettepe University, Ankara, Turkey

Keywords: Object Segmentation, RGB-D Data, Region Growing, Surface Normals.

Abstract: In this study, a method for automatic object segmentation on RGB-D data is proposed. Surface normals extracted from depth data are used to determine segment candidates first. Filtering steps are applied to depth map to get a better representation of the data. After filtering, an adapted version of region growing segmentation is performed using surface normal comparisons on depth data. Extracted surface segments are then compared with their spatial color similarity and depth proximity, and finally region merging is applied to obtain object segments. The method is tested on a well-known dataset, which has some complex table-top scenes containing multiple objects. The method produces comparable segmentation results according to related works.

1 INTRODUCTION

With the easy accessibility of powerful depth sensors, depth maps have become available as well as RGB images for any indoor scene. Along with these developments, scientific studies on 3D object recognition, segmentation, and tracking using RGB-D data have become popular in the recent years. In the field of robotic vision, depth sensors are used due to their low costs and mobility features.

Object detection and segmentation are very useful tasks for robotic grasping, as well as for the recognition and classification of the objects. Therefore, segmentation is an important step for robotic vision applications and used as an input for the next steps. In situations where segmentation is difficult with RGB information, more successful results can be achieved with RGB and depth information used together. Depth information enables to develop image processing approaches without being affected by color, texture and lighting features of the objects. Furthermore, depth information gives more cues about shape structures and contributes to a better understanding of the scene.

This paper introduces an automatic object segmentation method for tabletop scenes, which may contain mixed and complex sets with multiple objects in it. In this method, the segmentation process is performed in two main steps. Firstly, image segmentation is performed on depth data using an adapted version of region growing algorithm, based on surface normal similarity and smoothness. Then segmented regions are grouped using depth proximity and their RGB features based on color correlogram (Huang et al., 1997) and histogram similarity to obtain objects. In order to obtain better and accurate results, point cloud filtering steps are applied and smoothed depth data are obtained. The proposed method works unsupervised and automatically segments different objects in the scene. It produces meaningful segmentation results comparable with previous studies.

The outline of the paper is as follows: Section 2 gives the related works in the literature about object segmentation on RGB-D data. Section 3 introduces the proposed method. Section 4 gives the experimental results. Finally, Section 5 concludes with the given results.

2 RELATED WORK

Studies on image and object segmentation using RGB-D data has begun in the last seven years. Liu (Liu et al., 2013) studied automatic object segmentation using RGB-D cameras. In their camera with depth sensor (like Microsoft Kinect), there were

Yalic, H. and Can, A.
Automatic Object Segmentation on RGB-D Data using Surface Normals and Region Similarity.
DOI: 10.5220/0006617303790386
In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2018) - Volume 4: VISAPP*, pages 379-386
ISBN: 978-989-758-290-5

some gaps and absences in the depth map. This situation resulted in object boundaries not being properly determined. To solve this problem, they developed a three-way (trilateral) filter that includes distance, RGB values, and boundary information. They apply warping, error cleaning, and affine mapping to eliminate holes caused by depth sensor data acquisition. After this step, they used the probabilistic boundary detector (Pbd) component. Pb (probability of boundary) is a method that uses only color information and extracts the boundary-priority map of the object according to color differences. By adding depth attribute as a parameter, it calculates the probability that a pixel is at the boundary and its orientation. In the last step, they segmented the object using graph-cut and separated it from the scene. During the evaluation of their work, they measured the contribution of the depth information that they added to the basic methods. They have increased the performance of the segmentation.

Mishra (Mishra and Aloimonos, 2012, Mishra et al., 2012) developed a segmentation strategy that separates simple objects from the scene by using color, texture and depth knowledge. They defined the simple object as a compact zone surrounded by depth and contact boundary. They improved on the fixation-based segmentation method (Mishra et al., 2009), which they found the most suitable closed contour around the given point in the scene. In their recent work, they have proposed a fixing strategy that selects points from within an object, as well as a method that allows the selection of closed curves only for objects. The visual cues (color, depth) around each edge pixel indicate whether that pixel is at the object boundary. This information is kept in the probabilistic boundary edge map. An edge finder (Martin et al., 2004) with local brightness, texture and color cues, was used when calculating this map. If the segmentation process is examined sequentially, it is seen that first a probabilistic boundary edge map is obtained. By selecting the most probable edge pixels in this map, the object side is determined. By considering the object sides of the boundaries in this map, fixation points are selected on that side. Closed curves are obtained from the determined points by the fixed-based segmentation method and the resultant object segments are obtained. They have studied the experimental results in a comprehensive dataset (Lai et al., 2011), quantitatively and qualitatively. During quantitative analysis, they measured the segmentation accuracy of segmented objects as a single closed region and achieved a success rate of over 90%.

Richtsfeld and colleagues have performed numerous studies (Richtsfeld et al., 2012a), (Richtsfeld et al., 2014) for object segmentation on RGB-D data. In their study on the implementation of Gestalt principles for object segmentation (Richtsfeld et al., 2012b), they have defined the relationships between surface patches on a 3D image based on Gestalt principles in order to build a learning-based structure. The scene structure is rapidly abstracted by plane fitting on a 3D point cloud. The fast and commonly used RANSAC method is used for this purpose. But for curved objects, it is necessary to soften and bend the surfaces. At this point, they used the mathematical construct called NURBS (non-uniform rational B-splines), which is widely used in the field of graphics. This structure enables to display all kinds of conic sections (spherical, cylindrical, ellipsoid, etc.). The plane placed on the point cloud is matched to the cloud by minimizing the length of the nearest point. On the geometric structure, a final model selection is applied to determine the surface patches to be used in object segmentation.

In their article describing perceptual grouping for object segmentation on RGBD data (Richtsfeld et al., 2014), they presented a comprehensive study combining previous works. Once the surfaces have been determined, the relationships between adjacent surfaces, which are based on the Gestalt principles mentioned above, are calculated to group them. These features are surface color, the similarity of relative size and texture amount, color similarity at 3D surface boundaries, average curvature and curvature variance at 3D surface boundaries, average depth and variance at 2D surface boundaries. Two attribute vectors have been defined from the previous relations: neighbouring and non-neighbouring. These two vectors were used for training SVM on hand-labeled RGB-D image sets. The surface pairs of the same object are selected as positive samples, while the surface samples of two different objects or the object-background pairs are negative samples. In the decision making, SVM gives the probability value of each vector, as well as producing binary results. They defined a graph where surface patches represent the nodes and probability values obtained from SVM represents the edges. Finally using the graph-cut, object segmentation was performed. As a result of the detailed analysis of the segmented objects in different complex scenes separated by different categories, they achieved a success rate of over 90%. They stated that on the same data sets, Mishra et al. remained around 65%.

3 AUTOMATIC OBJECT SEGMENTATION

In this paper, an automatic method for segmenting objects in complex scenes on RGB-D data is proposed. To accomplish this, an image segmentation process is performed on the geometric structure of the objects using the depth information. Subsequently, object segmentation was performed using neighbourhood information based on the depth cue and color properties of the obtained regions.

The overall design of the system is illustrated in the Figure-1. Details of all steps will be given in the next sections of this paper.

3.1 Filtering

Depth data have some different features that are not present in its RGB image, depending on the depth acquisition device. Irregular density (describes number of points in a given area), noise, and outliers are some of the important ones. Previous studies (Liu et al., 2013) show that point cloud data needs to be filtered for better segmentation.

In our study, down-sampling like voxelgrid or morphological operations are not preferred because they modify the important properties of the point cloud set. We used filters to eliminate noises and remove outliers. The pass-through filter in Equation (1) used for eliminating the areas where the depth information is missing or larger than a pre-defined threshold. P denotes the point cloud set for x,y,z coordinates.

$$P' = \{P(x,y,z) \mid 0 < z < z_{threshold}\} \quad (1)$$

After that, radius outlier removal is used to erase outlier points. Let $n(x,y,z)$ be the number of neighbours for $P'(x,y,z)$ in a radius of r. We remove the points from the cloud with less than a given number of neighbours n_{min}within a radius r. In Equation (2), O denotes the outlier points set and F denotes the filtered points.

$$O = \{P'(x,y,z) \mid n(x,y,z) < n_{min}\}$$
$$F = P' - O \quad (2)$$

Pre-processing step is completed with filtering and suitable data is obtained for the segmentation step.

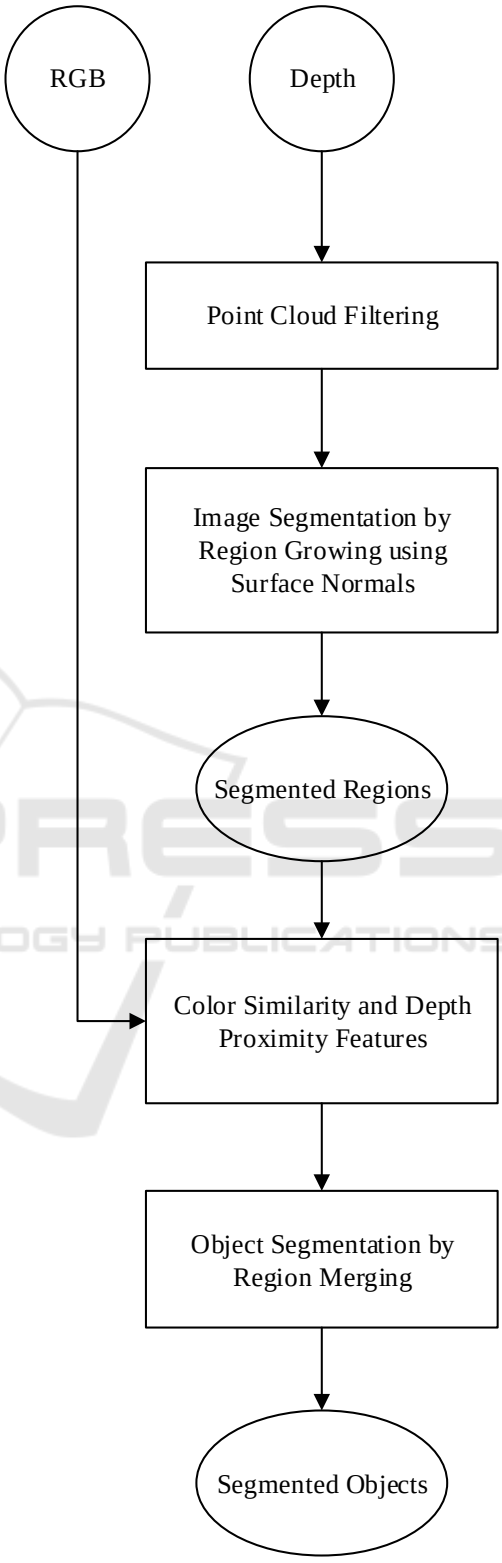

Figure 1: Overall system diagram for automatic object segmentation.

3.2 Region-growing based Segmentation using Normals

In our study, we used region-growing based segmentation to cluster depth data. Since we only deal with the depth data at this stage, we consider the angle difference between the surface normals of the points when growing the regions. Thus, while smooth surfaces are treated as a single region, regions with dramatically changing normals are separated into different segments.

3.2.1 Surface Normal Estimation

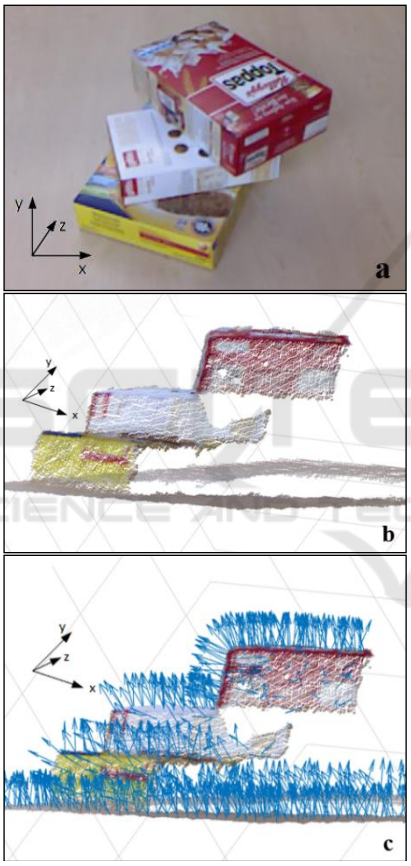

Figure 2: (a) Sample RGB image from the dataset (Richtsfeld et al., 2012a), (b) rotated depth image with surface mapping, (c) computed surface normals using depth map.

Surface normals are one of the important attributes of a geometric surface and widely used in computer graphics and vision applications. When a geometric surface is considered, it is usually insignificant to find a normal direction at a certain point on the surface, as a perpendicular vector to the surface at that point. However, since our acquired point cloud data form a set of point samples on the real surface, there are two common approaches for calculating the surface normal. The underlying surface can be obtained from the point cloud using surface mapping techniques and compute normals from the mesh. Another one is using approximations to derive the surface normal from the point cloud directly. We used the second approach since it gives sufficient surface normal vector of the points about belonging to the same surface.

We used widely known method (Hoppe et al., 1992) for surface normal estimation. Briefly, it selects a subset of points Φ inside point cloud P from the neighbourhood of p. Then it fits local plane through Φ and finally computes the normal vector N of the plane. Our method computes the normal vectors locally using six neighbouring points. The number of neighbouring points is optimal for our problem, and incrementing neighbour amount does not have any positive affect on segmentation according to our experiments. Figure-2 shows the illustrations of computed surface normals from the depth data of adjacent objects in the dataset.

3.2.2 Region-growing Segmentation

The aim of the algorithm is to merge similar points that are similar enough in terms of smoothness. Inputs of the algorithm are point cloud data after filtering, computed surface normal and their curvature value for each point in the cloud, normal angle threshold and curvature threshold. Outputs are the set of regions where each region is a set of points that are marked as a part of the same surface.

The algorithm is based on region-growing segmentation that is widely known image segmentation method. But the main difference is that it uses angle comparison between the point normals instead of intensity. A seed point is randomly selected and added to the set of seeds. The algorithm finds neighbouring points for every seed point. Every neighbour point is compared to the angle between its normal and current seeds normal. If it is less than the threshold ($10/180*\pi$ in radians, equals to 10 degrees), then the current point is added to the current segmented region. After that, every neighbour is compared for the curvature. If curvature value is less than the determined threshold (1.0), then this point is added to the set of seeds. Current seed is removed from the set. If the set of seeds becomes empty, that means the algorithm has grown the region and it is repeated from the beginning. As a result, we obtain object surfaces in the depth data that can be an important input for

object segmentation. Thresholds are determined according to data set calculations and tuned for optimal values. Pseudocode of the algorithm for region growing segmentation based on surface normal is shown below:

Inputs:
P // Points in the point cloud
N // Surface normal calculated from each point
C // Curvature value of the points
θ_{Nthres} // Normal angle threshold
C_{thres} // Curvature threshold
Set *SgtRgn* = { } // Segmented region list
Set *PointsToGrow* = *P* // Points available for growing
Algorithm:
While *PointsToGrow* != null do
| $SgtRgn_{current}$ = { } // Region segment
| $Seed_{current}$ ={ } // Set of current seeds
| *NextPoint* = rand (*PointsToGrow*)
| $Seed_{current} = Seed_{current} \cup NextPoint$
| $SgtRgn_{current} = SgtRgn_{current} \cup NextPoint$
| $PointsToGrow = PointsToGrow - NextPoint$
| For *i* = 0 to size($Seed_{current}$) do
| | *SeedCurrNeig* = FindNearNeigbours ($Seed_{current}[i]$)
| | For *j* = 0 to size(*SeedCurrNeig*) do
| | | P_j = *SeedCurrNeig*[*j*]
| | | If *PointsToGrow*.contains (P_j) AND
| | | cos-1($N[Seed_{current}[i]]$, $N[Seed_{current}[j]]$) < θ_{Nthres} then
| | | | | $SgtRgn_{current} = SgtRgn_{current} \cup P_j$
| | | | | $PointsToGrow = PointsToGrow - P_j$
| | | | | If $C[P_j]$ < C_{thres} then
| | | | | | $Seed_{current} = Seed_{current} \cup P_j$
| | | | | end if
| | | end if
| | end for
| end for
| $SgtRgn = SgtRgn \cup SgtRgn_{current}$
end while
Return *SgtRgn*

As a result of the algorithm, all the regions are obtained from the point cloud which are on the same object surface. Figure-3 shows the segmentation results after region growing which is mapped and compared with its original RGB image. As seen from the output, each surface of the 3D objects segmented as different regions. So, this yields to an over-segmentation problem that is one object is represented with multiple region segments. These regions need to be merged to obtain single object instances. There are also small regions segmented mostly in object boundaries because of surface normal values. These regions are also handled in the next region-merging section.

Figure 3: Region growing segmentation result.

3.3 Object Segmentation based on Spatial Color Similarity of the Regions

In order to achieve better segmentation performance, regions have to be merged appropriately for extracting objects from the scene. Only depth cue cannot be sufficient at this point. Because, as it can be seen in Figure 3, it is difficult to decide whether the boundary points belong to the same object or different object in complex scenes where there are multiple adjacent objects. Segmenting an object that is partially occluded by another object is relatively easy if there is a distance between them. For these reasons, in addition to depth information, we have proposed an approach that adds color information of regions to the process. Since the surfaces of the objects in the datasets have complex texture and color properties, using histograms did not yield successful results in our experiments.

To extract global color information of segments, we used Huang's approach (Huang et al., 1997), which is mostly known for image indexing or retrieval. It is a widely used image feature called color correlogram. It parses the spatial correlation of colors and tolerates large changes in appearance and shape caused by viewing directions. This property makes this feature convenient for our work because different surfaces of the same object have similar color characteristics.

In our method, the color image obtained from the region is firstly quantized as 64 colors (4x4x4) in RGB space. Then, using the 4 pre-defined distance values as in Huang's work, the neighborhoods at these distances are determined for each pixel. By taking the color values at these points, the correlogram is calculated and transformed into the 1x64 feature vector, resulting in a histogram-like color Auto-correlogram.

The color correlogram is calculated (denoted by $corr$) and sorted for all segmented regions. At the final stage, the color correlograms extracted from the regions and the contact boundaries of these regions have been taken into consideration in order to merge the regions. Starting from the first region, the correlogram vectors are compared ($dist$ is the histogram comparison function, squared Euclidean distance is used). And it is also checked whether these regions have contact boundaries. In doing so, all boundary points (denoted by $Bound$ function) of the region are compared to all boundary points of the other regions, and these two regions are merged if they share common boundary. ∩ operator specifies that two regions have mutual boundaries i.e.

boundary points are closer in means of depth distance. Below algorithm explains region merging process. These steps are performed iteratively until all regions are checked.

Algorithm:
$Object_0 = Segment_0$
For i = 0 to size(*Region_segments*) do
| For $j = i + 1$ to size(*Region_segments*) do
| | If ($Bound\ (Segment_i) \cap Bound\ (Segment_j) \neq \emptyset$) AND
| | $dist\ (corr\ (Segment_i), corr\ (Segment_j)) < thresh$ then
| | | $Object_x$ = Object of $Segment_i$
| | | $Object_x \leftarrow Segment_i \cup Segment_j$
| | end if
| end for
end for

After the object segmentation process is completed, there are still some areas which have few points, especially in the object boundaries. The correlogram extracted from these regions does not give meaningful results. Therefore, these regions are merged with another surface/region adjacent to them. This process, which has no effect on the segmentation quantitative results, has the effect on quality of the results and it provides a better representation of the segmented object. The region with the maximum number of points is not treated as an object because it represents the plane on which the objects are standing (mostly surface of the table).

4 EXPERIMENTAL RESULTS

In this section, experimental results of the proposed method are discussed. Experiments are performed on four datasets. First one is (Lai et al., 2011) RGB-D Objects Dataset. It consists of 300 objects in 51 categories. This set is widely used for object recognition, but we used it only for evaluation of our object segmentation method on single instance objects, not for comparison. They also have another dataset (Lai et al., 2012) called RGB-D Scenes Dataset which consists of 14 scenes and many objects with multiple views. The Object Segmentation Dataset (Richtsfeld et al., 2012a) contains 111 scenes which consist of boxes, cylindrical objects, occluded and stack objects with complex scenes. Willow garage dataset (Aldoma and Richtsfeld, 2012) contains approximately 160 frames includes mixed objects. All datasets are moderately realistic and contain table-top scenes with objects stand on it.

Table 1: Object segmentation accuracy on different scene categories compared with related works.

	Segmentation accuracy %		
Scene category	**Ours**	**Richtsfeld**	**Mishra**
Various single objects (dataset Lai,2011)	92	NA	NA
Single boxes	95	99	71
Cylindrical objects	90	99	66
Stacked boxes	90	93	64
Occluded objects	91	99	76
Mixed objects	88	94	62
Complex scenes	82	89	55

During quantitative analysis, we considered the number of objects segmented correctly from the total number of objects. We used sub-categories such as single object instances (contains various type of objects like apple, ball, cap, banana etc.), and also different categories described in (Richtsfeld et al., 2014) like single boxes, cylindrical objects, stacked boxes, occluded objects, mixed objects and complex scenes. Table-1 shows the object segmentation accuracy of our method on different scene categories compared with related works. Scene category of the various single objects are from dataset (Lai et al., 2011) and other categories are from dataset (Richtsfeld et al.). As expected, our method gives the best results in itself at the scenes where there is only one object. The segmentation of simple objects like boxes is more successful than objects with complex surfaces. Multiple stacked objects or occlusions do not have a negative affect on our segmentation performance. Also, the number of objects in a scene, would not affect performance, but accuracy is reduced in more complex scenes (Figure-4). When we examine fault cases of our method, we can see that different surfaces of the object are grouped incorrectly in cases where they contain different color information. Richtsfeld's method seems better than our method for many scene categories in Table-1, because they used learning techniques for relations between object surfaces, and that leads to better segmentation results in complex scenes.

One of the most important factors affecting the segmentation performance is the image resolution. Images in the dataset (depth and RGB) have a maximum resolution of 640x480 pixels. Due to the small size of the objects in the scene, the properties of the surface of the objects (like color, surface normals) cannot be extracted precisely. As a result, incorrect segmentation occurs.

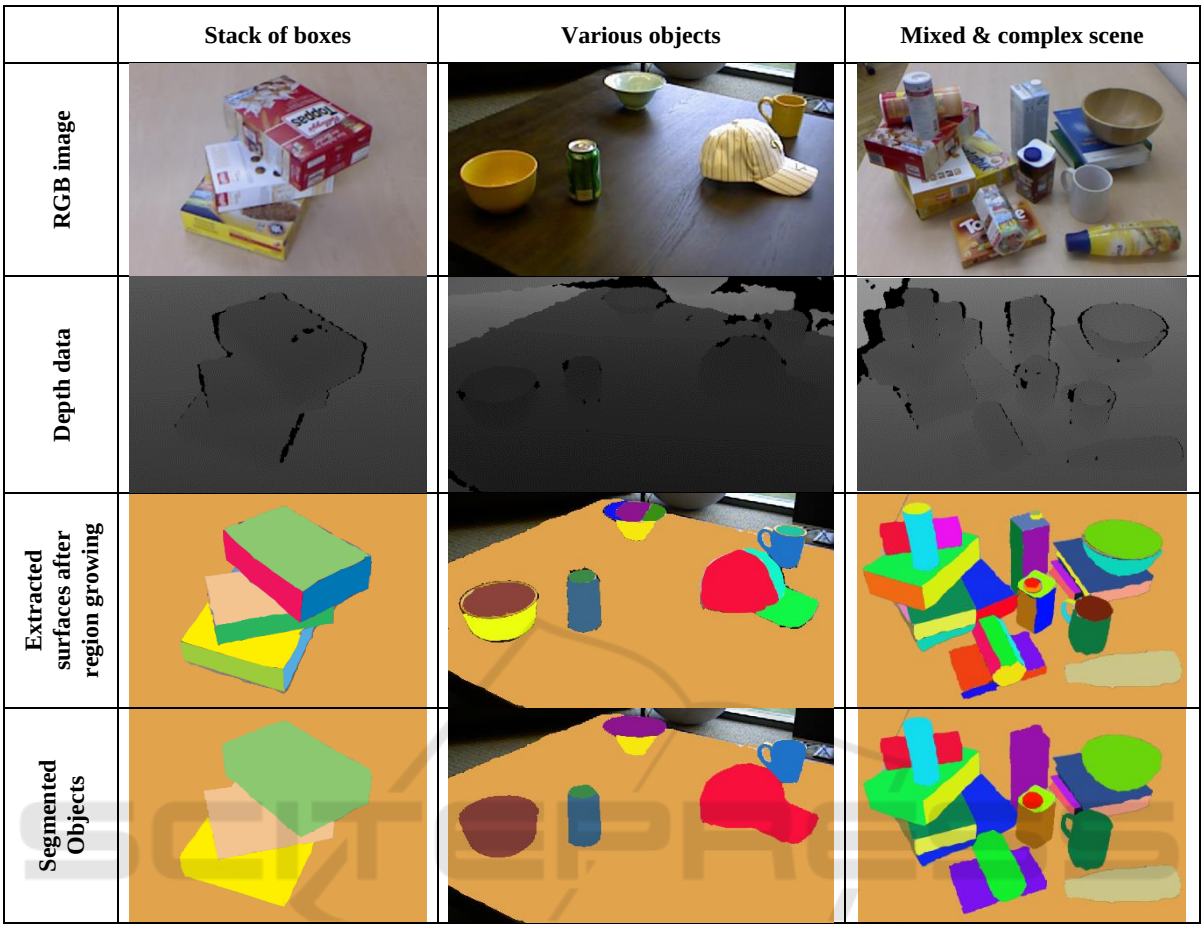

Figure 4: Sample results from the datasets (First column and last column images (Richtsfeld et al., 2012a), middle image (Aldoma and Richtsfeld)). Images in first two rows are from the datasets and last two rows are produced from our method.

When comparing the performance of our method with the related works, we considered those who experimented on the same dataset with us. In doing so, we have included the best segmentation accuracy of those studies that they mentioned. As it can be seen in Table 2, our method is competing with or even exceeding the relevant works in the field of RGB-D object segmentation. Our comparison also includes the results obtained by using only depth information, where the colors on the surfaces are not taken into consideration. Although we can achieve a certain performance using only depth information, we get the best results with color + depth data.

We compared the number of points difference between ground truth data and our segmented object while performing qualitative analysis. The ratio of the number of points in the object segment to the difference gives the incorrectly segmented points. The average value is 1.23% through all scenes in all of the datasets, that means segmented objects have 98.77% correct points compared to ground truth.

Table 2: Comparison of our method with the related works (overall percentage of objects segmented correctly).

Dataset	Methods			
	Richtsfeld	Mishra	**Ours only depth**	**Ours depth+color**
Richtsfeld – Object Segmentation Dataset (Richtsfeld et al., 2012a)	93	62	74	89
Willow garage dataset (Aldoma and Richtsfeld, 2012)	92	87	82	90
Lai – RGBD Scenes Dataset (Lai et al., 2012)	NA	91	80	93

5 CONCLUSIONS

We proposed an automatic object segmentation method using RGB-D data based on region growing and region similarity. We apply point cloud filtering for better abstraction of the depth data. After that region growing segmentation is performed on point cloud data based on surface normals. Since segmentation with surface normal causes over-segmentation, segmented surfaces of the objects need to be grouped at the final step. Thus, we apply a region merging method based on color correlogram similarity and depth proximity information of surface regions.

Experimental results show that our method has comparable segmentation accuracy. The contribution of our study is that primary features like surface normals and color similarity of object surfaces can be used for object segmentation using RGB-D data. We explained a method that works unsupervised and does not need to know how many objects exist in the scene. Additionally, objects category is not an issue.

As a future work, machine learning methods can be used to extract surface relations. Furthermore, computation time is high and thus the proposed method cannot be used in a real-time system for now. If the algorithm is parallelized by taking advantage of GPUs, the method can achieve real-time video processing.

REFERENCES

Aldoma, A. & Richtsfeld, A. 2012. *The willow garage object recognition challenge* [Online]. Available: http://www.acin.tuwien.ac.at/forschung/v4r/software-tools/object-instance-recognition-dataset/ [Accessed].

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. & Stuetzle, W. 1992. Surface reconstruction from unorganized points. *SIGGRAPH Computer Graphics*, 26**,** 71-78.

Huang, J., Kumar, S. R., Mitra, M., Zhu, W.-J. & Zabih, R. 1997. Image indexing using color correlograms. *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, IEEE, 762-768.

Lai, K., Bo, L., Ren, X. & Fox, D. 2011. A large-scale hierarchical multi-view rgb-d object dataset. *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, 1817-1824.

Lai, K., Bo, L., Ren, X. & Fox, D. 2012. Detection-based object labeling in 3d scenes. *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, 1330-1337.

Liu, H., Philipose, M. & Sun, M.-T. 2013. Automatic Objects Segmentation with RGB-D Cameras. *Journal of Visual Communication and Image Representation*.

Martin, D. R., Fowlkes, C. C. & Malik, J. 2004. Learning to detect natural image boundaries using local brightness, color, and texture cues. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26**,** 530-549.

Mishra, A., Aloimonos, Y. & Fah, C. L. 2009. Active segmentation with fixation. *Computer Vision, 2009 IEEE 12th International Conference on*, IEEE, 468-475.

Mishra, A. K. & Aloimonos, Y. 2012. Visual segmentation of "Simple" objects for robots. *Robotics: Science and Systems VII***,** 217.

Mishra, A. K., Shrivastava, A. & Aloimonos, Y. 2012. Segmenting "simple" objects using RGB-D. *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, 4406-4413.

Richtsfeld, A., Morwald, T., Prankl, J., Zillich, M. & Vincze, M. 2012a. Segmentation of unknown objects in indoor environments. *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, IEEE, 4791-4796.

Richtsfeld, A., Mörwald, T., Prankl, J., Zillich, M. & Vincze, M. 2014. Learning of perceptual grouping for object segmentation on RGB-D data. *Journal of visual communication and image representation*, 25**,** 64-73.

Richtsfeld, A., Zillich, M. & Vincze, M. 2012b. Implementation of Gestalt principles for object segmentation. *Pattern Recognition (ICPR), 2012 21st International Conference on*, IEEE, 1330-1333.