Image Segmentation from RGBD Images by 3D Point Cloud Attributes and High-Level Features

Mehdi Khazaeli mkhazaeli@pacific.edu
Department of Civil Engineering
University of the Pacific
Stockton, 95211, USA

Leili Javadpour ljavadpour@pacific.edu
Department of Computer Science
University of the Pacific
Stockton, 95211, USA

Gerald M. Knapp gknapp@lsu.edu
Department of Mechanical and Industrial Engineering
Louisiana State University
Baton Rouge, 70803, USA

Abstract

In this paper, an approach is developed for segmenting an image into major surfaces and potential objects using RGBD images and 3D point cloud data retrieved from a Kinect sensor. In the proposed segmentation algorithm, depth and RGB data are mapped together. Color, texture, XYZ world coordinates, and normal-, surface-, and graph-based segmentation index features are then generated for each pixel point. These attributes are used to cluster similar points together and segment the image. The inclusion of new depth-related features provided improved segmentation performance over RGB-only algorithms by resolving illumination and occlusion problems that cannot be handled using graph-based segmentation algorithms, as well as accurately identifying pixels associated with the main structure components of rooms (walls, ceilings, floors). Since each segment is a potential object or structure, the output of this algorithm is intended to be used for object recognition. The algorithm has been tested on commercial building images and results show the usability of the algorithm in real time applications.

Keywords: Graph-based Segmentation, Normals, RANSAC, Surface Detection, Occlusion.

1. INTRODUCTION

The invention of the low-cost Microsoft Kinect sensor has led to the availability of high-resolution depth and visual (RGB) sensing for wide areas of application [1]. The complementary nature of the depth and visual (RGB) information in the Kinect sensor opens up new opportunities to solve fundamental problems in segmentation, object recognition, and image retrieval problems.

Segmentation is concerned with identifying the parts of an image that are likely to correspond to individual objects. More precisely, image segmentation assigns a label to every pixel in an image such that pixels with the same label share certain visual attributes [2]. Segmentation is a precursor to object recognition and many other image processing applications.

There are significant challenges for segmentation, some of which follow:

- Varying weather/lighting conditions and varying exposure settings, each of which can change coloration and introduce or alter color gradients, shadows, and glare.
- Picture focus.
- Picture resolution.

- Viewpoint and zoom variations: Pictures of the same scene can be taken from different angles, heights, distances, and zoom settings.
- Occlusions: Color-only pictures are a 2D representation of a 3D world. Objects in the foreground occlude objects further back and may "split" background objects into multiple segments. Determining which segments go together is a significant computational task.

In this paper, we investigate – within the context of images of commercial office space – the effectiveness of using depth-related features of XYZ coordinates, surface normals, and surface planes in the segmentation process, as well as the effectiveness of more traditional color and surface features. We also develop a method for identifying those segments associated with the structural elements of rooms (walls, ceilings, floors). Finally, performance of the framework in terms of time and accuracy is analyzed.

2. LITERATURE REVIEW

In image processing, finding an object within an image is computationally expensive. The basic problem lies in detecting objects without knowing their identities in advance.

In recent research, regions were used for object detection instead of the traditional sliding window approach [3]. An assumption was made that each region represented a potential object or structure. Pont-Tuset et al. [3] develop a hierarchical segmentation algorithm that makes effective use of multiscale information. The also proposed a grouping strategy that combines the multiscale regions into object proposals. Liu et al. [4] used a non-parametric approach to image labeling by warping a given image onto a larger set of labeled images and then combining the results.

In the work done by Zlateski and Seung [5] implemented a hierarchical image segmentation algorithm that over-segments the image into watershed sections, defines a new graph on them, and then merges them with a modified, size-dependent version of single linkage clustering.

Blobworld segmentation [6] is another widely used algorithm, obtained by clustering pixels within a common color-texture-position feature space.

According to Liu et al. [4], discriminating between textures is the main difficulty of a segmentation method. Many texture segmentation algorithms require the estimation of texture model parameters – a very difficult task. 'JSEG' segmentation [7] overcomes this problem. Instead of trying to estimate a specific model per texture region, it tests for the homogeneity of a given color-texture pattern. 'JSEG' consists of two steps. In the first step, image colors are quantized to several classes. By replacing the image pixels with their corresponding color class labels, a class-map of the image is obtained. Spatial segmentation is then performed on the class-map, which can be viewed as a special type of texture composition. The algorithm produces homogeneous color-texture regions and is used in many systems [8].

Contour models or 'snakes' are one of the popular techniques for image segmentation. A snake is an energy-minimizing spline guided by external constraint forces and compelled by image forces that pull it toward features such as lines and edges. In other words, it recognizes nearby edges and localizes them accurately [9].

Region growing is a segmentation algorithm that starts with seeded pixels, and then adds neighboring pixels to those seeded region sets that are within a certain feature metric distance (e.g., within an intensity-level tolerance) [7]. There are also unseeded versions of the algorithm. Region growing algorithms are highly dependent upon the similarity metric used, which in turn is dependent upon the application.

Connected component labeling is a segmentation algorithm similar to region growing but employing a graph-based technique for segmentation. The basic premise of connected components is formation of regions containing pixels of the same value. Best results are obtained with a binary image. Connected components yields labeled segments that are the same in intensity and space, whereas an algorithm such as K-means yields those similar in intensity only. Graph partitioning considers each pixel a node and the distance to neighboring pixels as edges [10]. Edges that are more than a distance metric are removed and the image is then segmented.

Graph-based image segmentation techniques generally represent the problem in terms of a graph G = (V; E), where each node vi ϵ V corresponds to a pixel in the image, and the edges in E connect certain pairs of neighboring pixels [2]. Each edge is assigned a weight based on the properties of the pixels it connects (e.g., image intensities). Depending on the method, there may or may not be an edge connecting each pair of vertices. The earliest graph-based methods use fixed thresholds and local measures in computing segmentation [10]. Zahn et al. [11] introduced a segmentation method based on the minimum spanning tree (MST) of the graph. This method has been applied both to point clustering and image segmentation. For image segmentation, the edge weights in the graph are based on the differences between pixel intensities, whereas for point clustering, the weights are based on distances between points.

In recent years, researchers in computer graphics and image processing have focused much time and interest on extracting features using active sensors, stereo cameras, range scans, and time-to-flight cameras [1]. Laser scanning has also been used to extract depth features and find the point cloud; however, additional infrastructure is required and the method is time consuming to set up. Some of these technologies are expensive and cannot be used for all materials and objects. Kinect is an exciting, emerging alternative to these methodologies, providing high quality features comprised of color and depth, with real world applications in visual perception.

In the field of computer science, researchers have studied ways to label objects in an image by running machine-learning techniques on 3D point cloud data retrieved from both real world environments and from Google 3D Warehouse (a database of 3D models of objects). Google 3D Warehouse can then learn to recognize objects in the real world based on the shape of the model objects [12]. Richtsfeld et al. [13] developed a method for detecting 3D objects in RGBD-images and extracting representations for robotics tasks.

In work done by Newcombe et al. [15], a 3D indoor surface model was constructed. Rather than aligning the depth frames, they simultaneously estimated the camera poses and tracked live depth images against the constructed model. The reconstruction results were presented with low drifting and high accuracy, demonstrating the advantage of tracking against a global model versus frame-to-frame alignment.

Lai et al. [15] published an object dataset and highlighted the impact of depth information on object detection using histograms of oriented gradients (HOG) over color and depth images. They proposed an approach that segments objects from video using depth maps, and incorporates temporal information from optical flow to correct for motion artifacts in the depth data.

Another approach for indoor scene segmentation was proposed by Silberman et al. [16]. The methodology uses a CRF-based model to evaluate the range of different representations for depth information. They concentrated on finding main segmentations of the image and then studied the effect of the main segmentation on other segments.

For this research, modifications were made to the graph-based segmentation methods of Felzenswalb [17] by adding parameters and by mapping pixels of feature space and classifying them based on the similarities between feature vectors.

3. KINECT

The Kinect camera system (Figure 1) has both an RGB color camera and an infrared depth camera. The color camera supports a maximum resolution of 1280$\times$960 and the depth camera supports a maximum resolution of 640×480. The IR projector transmits a fixed pattern of light and dark speckles, which is reflected by objects and read by the IR camera. The RGB camera captures color images and the IR camera and the IR projector form a stereo pair with a baseline of approximately 7.5 cm.

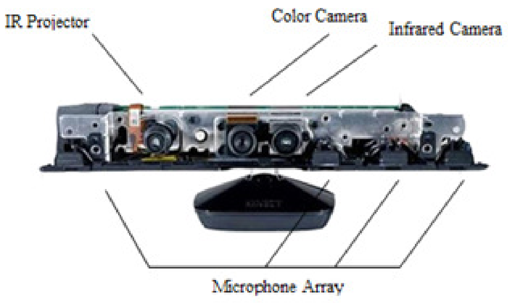

FIGURE 1: The Kinect Components.

Several free software development libraries are readily available (e.g., CLNUI, OpenNI, and Microsoft Windows SDK), enabling researchers to access and manipulate depth and color data via library calls. These platforms can be utilized in a broad range of applications.

Depth is calculated by triangulation against a known pattern from the projector. The reflected beam is converted into depth information by measuring the distance of the object in world coordinates relative to the sensor. A problem in using Kinect arises in the alignment of depth and color data. Also, the depth map is only valid for objects that are more than 2.5 feet and less than 13 feet away from the Kinect sensing device. If an object is closer than 2.5 feet or farther than 13 feet, its depth will not be detected correctly and will have a zero value.

In order to generate 3D points combining Kinect's color and depth cameras, the system must first be calibrated. This includes internal calibration for each camera (intrinsic) as well as relative pose and translation calibration between the two cameras (extrinsic). The Kinect device returns a processed image that is not aligned with the original infrared image. The best match gives an offset from the known depth in terms of pixels. For each pixel, the distance to the sensor can be obtained by triangulation.

Our setup consists of a depth and a color camera, both of which are calibrated simultaneously by finding the intrinsic parameters (focal length, principal point, distortions) for each camera.

Mapping depth and RGB pixel data together requires extrinsic calibration to "learn" these differences and the translations. Calibration is performed using the RGBDemo toolbox [18]. A chessboard of pattern size 0.25 cm is printed and glued to a board. The toolbox is used to grab images from the chessboard while simultaneously calibrating the color and depth cameras, as well as the relative pose between them. The toolbox extracts the checkerboard corners from the image and computes a homography using the known corner positions in world coordinates and the measured positions in the image. Figure 2 shows the images with the detected corners.

In this way, the intrinsic camera matrixes, camera calibration matrix, and distortion parameters of the two cameras are obtained. The geometrical relationship between the IR and color cameras is computed by retrieving extrinsic parameters from the calibration toolbox. Specific calibration is explained in detail at the RGBDemo website [18].

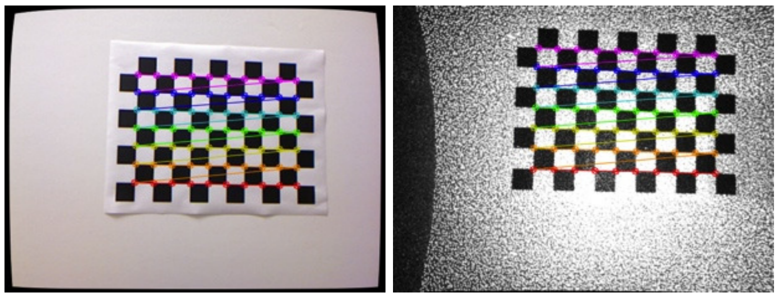

FIGURE 2: Left: Calibration of color camera; Right: Calibration of infrared camera.

4. METHODOLOGY

Segmentation is performed using a feature-based clustering algorithm. Four main groups of features are used. One group consists of color and texture features based only on the RGB camera data. The second group comprises depth-based features, specifically XYZ world coordinates and pixel normal data. A third feature is formed by using the merged RGB and depth data and applying the RANSAC (RANdom SAmple Consensus) algorithm [19] to find a plane index that best fits each pixel and its neighborhood. The final feature is built upon Felzenswalb and Huttenocher's [17] graph-based image segmentation. The original algorithm uses only color features to segment the image. This algorithm has performed well relative to other RGB segmenters, but cannot handle occlusion. For this reason, it was used as a preprocessor to generate a nominal segment feature for input to our classification model. In addition, to improve performance of the Felzenszwalb and Huttenlocher model, the algorithm has been modified to consider geometric (depth-related) information at each pixel point in its cost function. A surface index feature is used to help overcome occlusion. In the proposed segmentation algorithm, bottom up features are combined with top down features to satisfy pixel level and other high-level properties of an image. Figure 3 summarizes the overall process; details are provided in the following sections.

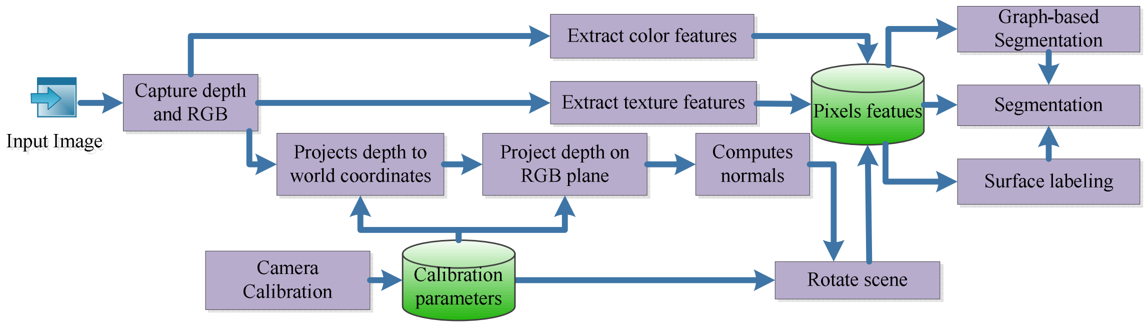

FIGURE 3: Segmentation Methodology.

4.1 Pixel Features

The calibration parameters are used for projecting depth onto the RGB plane and finding XYZ world coordinates. Raw depth values are integer values between 0 and 2047. They can be transformed into depth in meters with a linear model supplied by the RGBDemo toolbox [18]. After calculating depth in meters, the following formula is used for converting each pixel in the RGBD image to XYZ world coordinates.

$$x = (i - w/2) \times (z - 10) \times 0.0021 \times (w/h)$$

$$y = (j - h/2) \times (z - 10) \times 0.0021$$

$$z = z$$

where i and j are the location of a pixel in the 2D image; w and h are width and height of the 2D image; z is the depth in meters for each pixel captured from the depth camera.

These parameters are valid for all current Kinect sensors and are given on the OpenKinect webpage (http://openkinect.org/wiki/Imaging_Information). Figure 4 depicts the 3D reconstruction of points in space for the picture on the left.

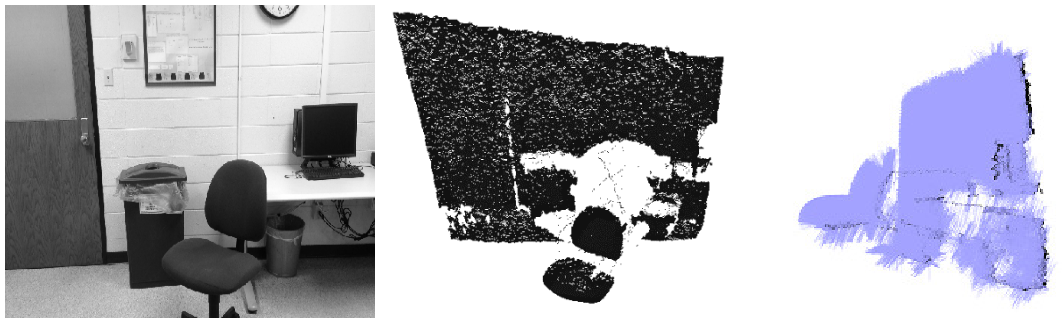

FIGURE 4: Left image: original image; middle image: shows XYZ in world coordinates; right image: shows normal vector at each point.

The pixel normal is a unit vector perpendicular to the surface at a given point. The normal for each pixel in the image is approximated by estimating the normal of a plane tangent to the pixel and its neighboring points. This is a least-square plane fitting estimation problem solved by using the eigenvectors and eigenvalues of a covariance matrix created from the nearest neighbors of the query point. After finding the first three eigenvalues, the values are normalized to produce the normal representation for the point. A matrix of surface normals is generated for each point (p) in the 3D point cloud by locating the nearest neighbors of p and computing the surface normal (n) of each point.

To extract a surface normal at each pixel, planes were fitted to 3D point clouds. The equation of the plane is given by:

$$n_x \times p_{ix} + n_y \times p_{iy} + n_z \times p_{iz} + d = 0$$

where $n = [n_x \quad n_y \quad n_z]$ is normal orientation for a given point cloud; pix, piy and piz are the XYZ world coordinates for point I; and d is the plane offset.

To determine the relative orientation of the camera with respect to the scene, the vanishing points are identified. For commercial office space, the majority of lines are parallel or orthogonal to one another. These lines, when projected onto the image plane, meet at the vanishing points. Detecting the vanishing points, therefore, aids in the understanding of an image. Vanishing point detection methods rely on line segments detected. Line segments are first clustered according to a shared common vanishing point obtained by projection of the intersection points of all pairs of line segments in the image onto the Gaussian sphere. Then the dominant clusters of line segments are determined by considering the orthogonality criterion, the camera criterion, and the vanishing line criterion [20].

Once the vanishing points have been detected, the relative orientation of the camera with respect to the scene can be computed [21]. The rotational geometrical relationship between the depth and color cameras, and relative orientation of the camera with respect to the scene are shown in Figure 5.

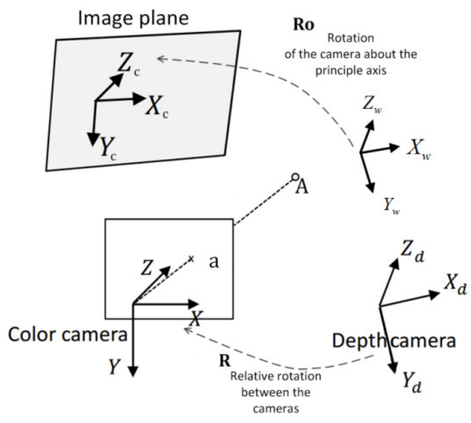

FIGURE 5: Rotation of the camera and relative rotation between the cameras.

To extract texture features, the image is converted into an HSV (Hue-Saturation-Value) color image. The V component has the widest range (in bits) and contains the most texture information. The grey level co-occurrence matrix is extracted from the V component. Contrast (a measure of intensity contrast between a pixel and its neighbor over the whole image), correlation (a measure of how correlated a pixel is to its neighbor over the whole image), energy (the sum of squared elements in the GLCM), and homogeneity (a value that measures the closeness of the distribution of elements in the GLCM to the GLCM diagonal) are calculated to generate texture features [Matlab toolbox]. By using the color histogram, similarity of color features is specified by counting the color intensities. All colors can be reproduced by combining the three primary colors (R, G and B). These three colors are represented as vectors in 3D RGB color space. For each pixel, 3 color features, 4 texture features, 3 features for normal and 3 features for location in the world coordinate system are extracted.

4.2 Surface Index

The main surfaces in the image are identified using the RANSAC (RANdom SAmple Consensus) algorithm detailed in [19] A surface index is assigned to each pixel in the image, and that index is then used as a feature in the segmentation process. RANSAC identifies points belonging to the same plane in space, taking into consideration that, for each pixel, the XYZ point should fall close to the plane, and the normals of each pixel should be close to the normal of the plane. A point is called an "inlier" if the distance from the point to the plane is smaller than a certain threshold. The threshold is the maximum perpendicular distance of a point to the plane. This distance is calculated based on the number of inliers over total number of points.

First, a small set of random sample points is selected. For each of these sample points, a group of neighbors is selected in horizontal and vertical directions and potential planes are generated using a RANSAC procedure for each sample point. At this point, many planes are generated. After a few iterations, the algorithm terminates and the final plane parameters are calculated. If the number of inliers is greater than the minimum number of inliers, the planes are selected as final surfaces and assigned an index value. A threshold of 1,000 for the minimum number of inliers was used; this value was found by finding the average total number of pixels with known depth in a set of images, and taking 0.5% (as specified in the RANSAC research). The algorithm then tries to merge surfaces with the same parameters together; pixels outside these surfaces are ignored. Figure 6 shows the surfaces selected using RANSAC.

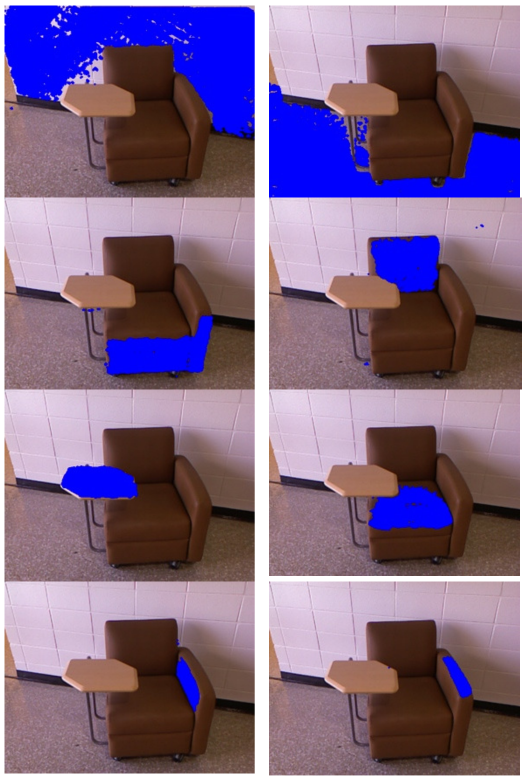

FIGURE 6: Surface Detection for a Sample Image.

As shown in Figure 7, the surface index algorithm can manage the occlusion problem. Although the wall is occluded by a cable management column and there are no neighboring pixel points between the divided parts of the wall, the system is able to include them in the same surface.

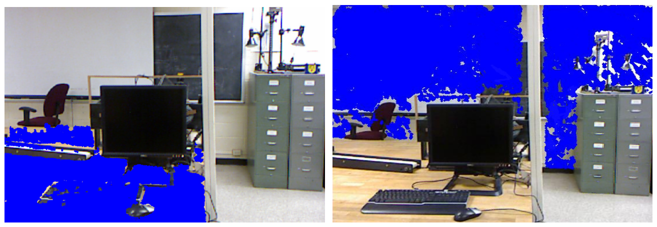

FIGURE 7: Occlusion Solved using Surface Index Algorithm.

Errors in surface assignment may arise. Figure 8 shows errors in surface detection when points in different planes of the same elevation are grouped together. In the final segmentation process, these errors are handled using color, texture, normal and XYZ world coordinate features.

A surface index is generated for each pixel point that corresponds to the surface in which the point is embedded. For points for which no surface was found, a random unique surface index was assigned. This strategy helps prevent these points from getting lumped together as a single surface by the classification algorithm. Surface index is a nominal feature used in the final segmentation stage.

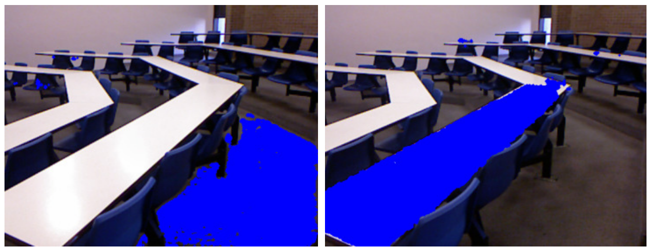

FIGURE 8: Errors in Surface Detection.

4.3 Graph-Based Region Index

The graph-based segmentation algorithm of Felzenszwalb and Huttenlocher was used due to higher accuracy yields as well as compatibility with our model. This approach has been found to be one of the most efficient segmentation algorithms currently available, yielding satisfactory segmentation results in real time [23]. Graph-based methods utilize a top-down approach, modeling the images as weighted graphs and providing segmentation by recursively partitioning the graph into sub-graphs. Felzenszwalb and Huttenlocher delineated the logic for measuring the evidence for a boundary between two regions. The algorithm, written in C++, is open-source and available from http://www.cs.brown.edu/~pff/segment. Input to the algorithm is individual images in PPM (Portable PixMap) format. Output is also in PPM format, with region pixels color-labeled. The original algorithm only uses color features to segment the image. Figure 9 shows examples of the graph-based segmentation proposed by Felzenszwalb and Huttenlocher.

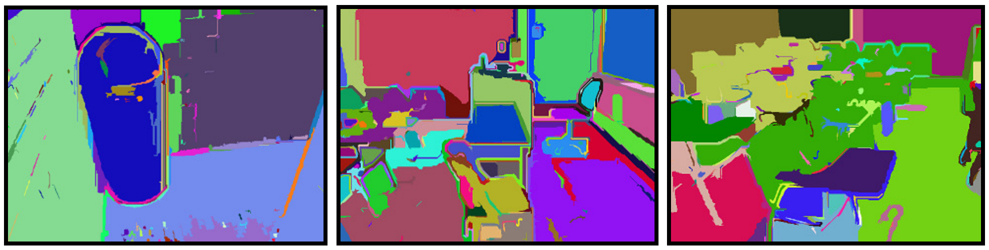

FIGURE 9: Segmentation using Felzenszwalb and Huttenlocher's Algorithm.

The limitations of the graph-based approach lie in handling occlusion. As shown in Figure 9, the floor has been segmented into different regions. Since this algorithm is based on intensity of neighboring pixels, illumination and shadow will affect its performance.

The proposed algorithm for object recognition is intended to be used in such a way that each segment corresponds to a potential object or structure. Therefore, to improve the performance, the original algorithm has been modified by adding depth information to the cost function. Graph-based segmentation is based on mapping each pixel to a point in feature space and finding clusters of similar points. The graph G(V,E) has vertices corresponding to each pixel in feature space and edges (vi,vj) connecting pairs of neighboring feature points vi and vj. Each edge has a weight w(vi,vj) that measures the dissimilarity between pixel points vi and vj. The edge weight is a function of the difference between intensity, distance, texture, motion, or any local attribute of two pixel points. TEdges between two pixel points in the same region should have low weights due to similarity of features, and edges between two pixel points in different regions should have higher weights. Felzenswalb and Huttenlocher's [17] edge weight is based on the location of the pixel in the image and the color value of the pixel. We modified their work by adding depth and normals, using L2 (Euclidean) distance between points. Normals are an important attribute of each pixel and reveal information on the orientation of each point. The direction of the normal at each point is the vector perpendicular to the surface at that point. This information can help segment regions with similar geometrid surface. The updated cost function is presented in the following equation:

$$w(v_i, v_j) = \omega_1 (|I(p_i) - I(p_j)|) + \omega_2 (\sqrt{(p_{ix} - p_{jx})^2 + (p_{iy} - p_{jy})^2 + (p_{iz} - p_{jz})^2}) +$$
$$\omega_3 (|Nx(p_i) - Nx(p_j)| + |Ny(p_i) - Ny(p_j)| + |Nz(p_i) - Nz(p_j)|)$$

where I and j are two neighboring pixel points; pix, piy, piz are the XYZ of point i in world coordinates; Nx, Ny, and Nz correspond to the normal at the pixel pi; {ω1, ω2, ω3} are normalization coefficients; and I(pi) is the intensity of the pixel pi.

The original algorithm was also modified to generate output in the form of a labeled segment index value for each pixel. Figure 10 shows the modified graph-based segmentation for a sample image.

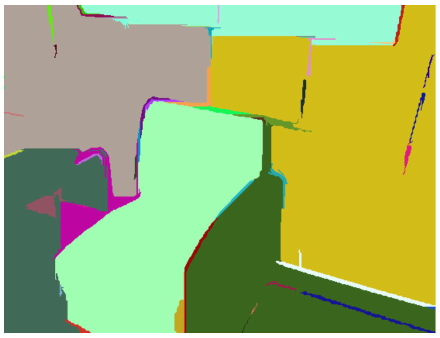

FIGURE 10: Graph-based Segmentation.

At this stage a graph-based region index is generated for each pixel point. This value is used as a nominal feature for segmentation.

In the following section, the graph-based region index is used along with surface index and pixel features in a clustering method to segment the image into regions of potential objects.

5. CLUSTERING

In previous sections, a surface index was generated for each pixel. The relative orientation of the camera with respect to the scene was computed using vanishing points. Hence each pixel point can now be transformed to world coordinates.

In this way the points that are inliers on the floor or ceiling have a z-normal of close to 1. For the inliers on the wall, depending on the direction of the wall, x-normal or y-normal has a value close to 1. If a surface has a normal component close to 1, and the minimum number of inliers is greater than the threshold, it is determined to be a structure in the image. Therefore these points are extracted from the image and segmentation is not applied to them.

For the remaining points, WEKA (Waikato Environment for Knowledge Analysis) [22] is used for applying an unsupervised classification method. To perform clustering, feature vectors were extracted for each pixel in the image. A feature vector consists of the following 15 features:

- 3 color features
- XYZ world coordinates
- texture features
- 3 normal features

- Surface index
- Graph-based segment ID

The data set consists of 105 multi-object images taken in different indoor scenes. The structure of the scenes (walls, floor, ceiling) are specified and the corresponding pixels' intensity is set to 0 to avoid re-segmentation. For the remaining parts of the image, unsupervised classification is used to cluster similar data together to form potential object regions; K-mean clustering with 5 clusters is used. If 'k' is increased, the number of segments is increased. If the number of iterations is increased, the accuracy is increased. A tradeoff exists between time and accuracy and therefore 500 iterations was used.

The result of this segmentation is a cluster ID for each pixel in the image. The pixels in the same cluster form regions that are considered to be a potential object.

The time breakdown for different stages of the process is shown in Table 1.

Stage	Time
Extracting pixel features (color, texture, normal, surface ID and XYZ)	84.9
Extracting graph-based segment ID	4.3
Clustering	3.4
Total	***92.6***

TABLE 1: Time Breakdown.

The total processing time is 92.6 seconds for each image. The majority of time is used to determine vanishing points. The rotation matrix 3D direction of each line is computed using SVD to find the direction of maximum variance. Determining the highest score principle directions is a time consuming process, but the time can be improved significantly using at least 4 GB RAM and an Intel dual core processor. The use of scripts in the program also increases run time.

Post-processing of the image involves applying a median filter to merge small clusters with the bigger clusters to which they are a part. The goal is finding potential objects and structures in the image. These smaller clusters can also be the error caused by undefined depth points in Kinect. In Figure 11, the image on the left is the segmented output of the framework and the image on the right shows the segmentations after applying a median filter.

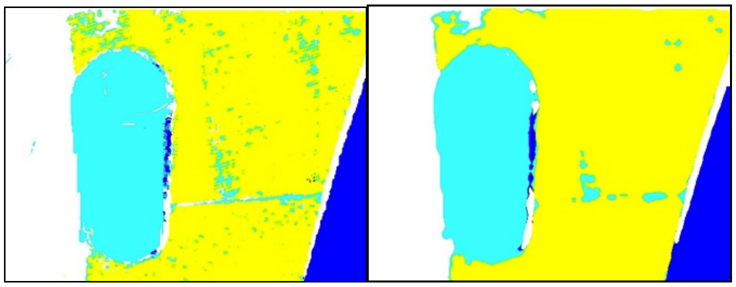

FIGURE 11: Segmentation result before and after applying median filtering.

6. CONCLUSIONS AND FUTURE WORK

The new segmentation methodology developed in this research modifies Felzenswalb and Huttenlocher's [17] cost function and combines the result with a new set of extracted features (surface index, depth, color, texture, XYZ world coordinates, and normals) used in a clustering method to overcome issues of occlusion and shadow.

After clustering, each image is divided into potential object regions and structures. Examples of the segmentation are shown in Figure 12. The segmented images reveal important global characteristics, structure and detail at the pixel level. Evaluating segmentation is based on three factors: precision, accuracy and efficiency. This technique runs close to real time and can be used as a preprocessing step in object recognition and therefor is efficient. To access accuracy we need to choose a surrogate of true segmentation and proceed as for precision.

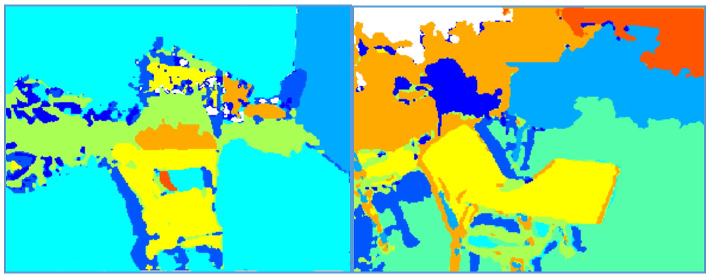

FIGURE 12: Image Segmentation.

Future work will first focus on implementing an algorithm to evaluation the performance of the segmentation system. Then it will focus on recognizing objects in each of the regions and incorporating video. Rules in video processing help to distinguish objects from their backgrounds. The work will be divided in two main sections (1) improving the filters used in the post-segmentation process resulting in rigid segments, and (2) the removal of tiny polygons and distortions that are not part of the object.

7. REFERENCES

[1] S. Izadi, D. Kim, O. Hilliges, D. Molyneauz, R. Newcombe, P. Kohli, J. Shotton, S. Hodege, D. Freeman, A. Davison, A. Fitzgibbon. “KinectFusion: Real-Time 3D Reconstruction and Interaction Using a Moving Depth Camera”. Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, 2011 pp. 559-568.

[2] K. K. Singh and A. Singh. “A Study of Image Segmentation Algorithms for Different Types of Images”. IJCSI International Journal of Computer Science, vol. 7, pp. 414-417, Sep. 2010.

[3] J. Pont-Tuset, P. Arbelaez, J. T. Barron, F. Marques, and J. Malik. (2015, Mar.) “Multiscale Combinatorial Grouping for Image Segmentation and Object Proposal Generation”. In arXiv preprint arXiv:1503.00848.

[4] C. Liu, J. Yuen and A. Torralba. “Nonparametric scene parsing: Label transfer via dense scene alignment”. Computer Vision and Pattern Recognition, 2009, pp. 1972-1979.

[5] A. Zlateski, and H. S. Seung. (2015, May). “Image Segmentation by Size-Dependent Single Linkage Clustering of a Watershed Basin Graph”. In arXiv preprint arXiv:1505.00249.

[6] C. Carson, S. Belongie, H. Greenspan, and J. Malik. “Blobworld: Image Segmentation Using Expectation Maximization and Its Application to Image Querying”. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, pp. 1026-1038, Aug. 2002.

[7] M. Paulinas, and A. Ušinskas. “A survey of genetic algorithms applications for image enhancement and segmentation”. Information Technology and control, vol. 36, pp. 278-284, Apr. 2015.

[8] H. Feng and T. S. Chua, “A Bootstrapping Approach to Annotating Large Image Collection”. 5th ACM SIGMM International Workshop on Multimedia Information Retrieval, 2010, pp. 55-62.

[9] C. H. Chen, L. Potdat, and R. Chittineni, “Two Novel ACM (Active Contour Model) Methods for Intravascular Ultrasound Image Segmentation”. AIP Conference Proceedings, 2010, pp. 735-741.

[10] J. Shi and J. Malik. “Normalized Cuts and Image Segmentation”. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, pp. 888-905, Aug. 2000.

[11] C. T. Zahn. “Graph-Theoretic Methods for Detecting and Describing Gestalt Clusters”. IEEE Transactions on Computing, vol. 20, pp. 68-86, Jan. 1971.

[12] K. Lai and D. Fox. “Object Recognition in 3D Point Clouds Using Web Data and Domain Adaptation”. International Journal of Robotics Research, vol. 29, pp. 1019-1037, Jul. 2010.

[13] A. Richtsfeld, T. Mörwald, J. Prankl, J. Balzer, M. Zillich, and M. Vincze. “Towards Scene Understanding–Object Segmentation Using RGBD-Images”. In Proceedings of the 2012 Computer Vision Winter Workshop (CVWW), 2012.

[14] R.A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A.w J. Davison, P. Kohli, J.e Shotton, S. Hodges, and A. Fitzgibbon. “KinectFusion: Real-Time Dense Surface Mapping and Tracking”. In Mixed and augmented reality (ISMAR), 10th IEEE international symposium, 2011, pp. 127-136.

[15] K. Lai, L. Bo, X. Ren, and D. Fox. “A Large-Scale Hierarchical Multi-View RGB-D Object Dataset”. International Conference on Robotics and Automation, 2011, pp. 1817-1824.

[16] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. “Indoor Segmentation and Support Inference from RGBD Images”. ECCV Proceedings of the 12th European Conference on Computer Vision, 2012, pp. 746-760.

[17] P. F. Felzenszwalb and D. P. Huttenlocher. “Efficient Graph-Based Image Segmentation”. International Journal of Computer Vision, vol. 59, pp. 167-181, Sep. 2004.

[18] N. Burrus, “RGBDemo,” http://rgbdemo.org/, 2011 [Aug. 4, 2014].

[19] R. C. Bolles, and M. A. Fischler. “A RANSAC-Based Approach to Model Fitting and Its Application to Finding Cylinders in Range Data”. International Joint Conference of Artificial Intelligence, v1981, pp. 637-643.

[20] C. Rother. “A New Approach to Vanishing Point Detection in Architectural Environments”, Image and Vision Computing, vol. 20, pp. 647-655, Aug. 2002.

[21] J. Kosecka and W. Zhang. “Video Compass”. In ECCV, Berlin, Springer, 2002, pp. 476-491.

[22] R. R. Bouckaert, E. Frank, M. A. Hall, G. Holmes, B.d Pfahringer, P. Reutemann, I. H. Witten. “WEKA-Experiences with a Java Open-Source Project”. The Journal of Machine Learning Research, vol.11, pp. 2533–2541, Mar. 2010.

[23] D. Narayan, S. K. Murthy, and G. H. Kumar. “Image Segmentation Based on Graph Theoretical Approach to Improve the Quality of Image Segmentation”. World Academy of Science, Engineering and Technology, vol. 42, pp.35-38, 2008.