Efficient Hierarchical Graph-Based Segmentation of RGBD Videos

Steven Hickson[1] shickson@gatech.edu
Stan Birchfield[2] stanleyb@microsoft.com
Irfan Essa[1] irfan@cc.gatech.edu
Henrik Christensen[1] hic@cc.gatech.edu
[1]Georgia Institute of Technology, Atlanta, GA, USA , [2]Microsoft Robotics, Seattle, WA, USA
http://www.cc.gatech.edu/cpl/projects/4dseg

Abstract

We present an efficient and scalable algorithm for segmenting 3D RGBD point clouds by combining depth, color, and temporal information using a multistage, hierarchical graph-based approach. Our algorithm processes a moving window over several point clouds to group similar regions over a graph, resulting in an initial over-segmentation. These regions are then merged to yield a dendrogram using agglomerative clustering via a minimum spanning tree algorithm. Bipartite graph matching at a given level of the hierarchical tree yields the final segmentation of the point clouds by maintaining region identities over arbitrarily long periods of time. We show that a multistage segmentation with depth then color yields better results than a linear combination of depth and color. Due to its incremental processing, our algorithm can process videos of any length and in a streaming pipeline. The algorithm's ability to produce robust, efficient segmentation is demonstrated with numerous experimental results on challenging sequences from our own as well as public RGBD data sets.

1. Introduction

Segmentation is used as a building block for solving problems such as object detection, activity recognition, and scene understanding. Traditionally, segmentation involves finding pixels with similar perceptual appearance in a *single image* and grouping them. In recent years, there has been increasing emphasis upon *video segmentation*, in which pixels with similar appearance and spatiotemporal continuity are grouped together over a video volume[11, 7, 23]. Given the widespread availability of depth data from low-cost RGBD sensors we are interested in robust unsupervised segmentation of *streaming 3D point clouds* with temporal coherence. Our algorithm is aimed at *super-toxel* extraction; since voxels are limited to 3D data, we define a new term *toxels*. Toxels are *temporal voxels*, which can be understood as a generalization of voxels for 4D hyperrectangles. Each toxel is a hypercube discretization of a continuous 4D spatiotemporal hyperrectangle. Voxels are generally used to refer to the discretization of 3D volumes or the discretization of 2D frames over time. When these are combined, we get XYZT data, which can be discretized using toxels. Super-toxels are just large groupings of toxels.

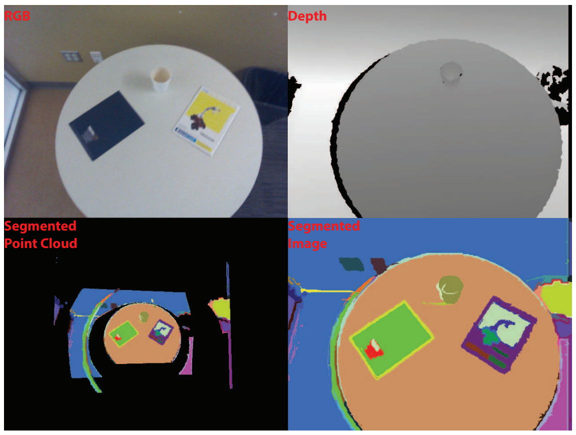

Figure 1. Segmentation of streaming 3D Point clouds. Top: RGB and depth maps used as input in our algorithm. Bottom: Rendered version of the point cloud and its frame segmentation output.

Our method uses measurements such as color, spatial coordinates, and RGBD optical flow to build a hierarchical region tree for each sequence of n frames (we use $n = 8$), which are then sequentially matched to produce long-term continuity of region identities throughout the sequence. This bottom-up design avoids limitations on the length of the video or on the amount of memory needed due to the high volume of data. In contrast to segmentation-tracking methods, we do not assume *a priori* knowledge of the content of the scene and/or objects, and we do not perform initialization of seeds or any other type of human intervention. Our motivation is that regions should not merge across depth boundaries despite color similarity. Our method was tested on several challenging sequences from the TUM Dataset [20], the NYU Dataset [18], and our own examples. It produces meaningful segmentation, even under difficult conditions such as significant motion of scene objects, camera motion, illumination changes, and partial occlusion.

Our primary contributions are: (1) A robust and scalable, RBGD video segmentation framework for streaming data. (2) A streaming method that maintains temporal consistency and can be run on a robot or on videos of indefinite length. (3) An efficient framework that runs at 0.8 fps but can be downsized to run pairwise in near real-time at 15 fps. (4) A nonlinear multistage method to segment color and depth that enforces that regions never merge over depth boundaries despite color similarities. And finally (5) The ability to segment and maintain temporal coherence with camera movement and object movement. The code and data are publicly available at the paper web page.

2. Related Work

One way to organize related work is use the five types of algorithms for super-voxel video segmentation analyzed by Xu and Corso [23]. (A) Paris and Durand [11] propose a *mean shift* method that achieves hierarchical segmentation in videos using topological persistence using the classic mode-seeking meanshift algorithm interpreted under Morse theory as a topological decomposition of the feature space. (B) [17] use *Nystrom normalized cuts*, in which the Nystrom approximation is applied to solve the normalized cut problem, for spatiotemporal grouping. (C) *Segmentation by Weighted Aggregation (SWA)* [16] is a variant of optimizing the normalized cut that computes a hierarchy of sequentially coarser segments by an algebraic multigrid solver. (D) *Graph-Based (GB)* is an adaptation of the Felzenszwalb and Huttenlocher image segmentation algorithm [5] to video segmentation by building the graph in the spatiotemporal volume where voxels (volumetric pixels) are nodes connected to 26 neighbors. (E) *Hierarchical graph based (GBH)* is an algorithm for video segmentation proposed in [7] that iteratively builds a tree structure of region graphs, starting from over-segmented spatiotemporal volumes obtained using the method illustrated above. The regions are described by LAB histograms of the voxel members, the edge weights are defined by the χ^2 distance, and the regions are merged using the same technique as in [5].

Surveying the literature, there has been little work on unsupervised segmentation of streaming 3D point clouds besides [1], which uses a parallel Metropolis algorithm and the Potts model and focuses only on a specific environment and [22], which combines color and depth and uses spectral graph clustering without a hierarchical method. We are unable to compare against either of these since no code or data was released. Additionally, most of the research conducted in 3D segmentation of point clouds has been focused on static scenes, without considering time. For example, planar and non-planar surfaces are segmented from 3D point clouds using either NURBS [8, 14], surface normal segmentation [19], or 3D shape matching [10] to identify objects with a particular shape. Scene modeling is performed by analyzing support relationships of the regions [18] or contextual modeling of both super-pixel MRFs and paths in segmentation trees [13]. Temporal evolution of the 3D point cloud has been considered in cases where a learned 3D model of the segmented object is available, such as in the simultaneous segmentation and pose tracking approach of [12], the rigid object tracking approach of [15], or the segmentation-tracking method of an arbitrary untrained object in [21].

3. Proposed Method

Our approach involves four main steps, as illustrated in Figure 2. First, a segmentation of 8 consecutive frames using the depth and motion information is performed. In the second step, an over-segmentation of the frames is done using color and motion information while respecting depth boundaries. Next, histograms of the resulting regions are used to build a hierarchical segmentation of the spatiotemporal volume represented as a dendrogram, which can then yield a particular segmentation depending on the desired segmentation level output. The final step performs a bipartite graph matching with the 8 previous frames with 4 frames overlapping to enforce the consistency of region identities over time.

3.1. Spatiotemporal segmentation

Graph-based segmentation: A natural approach to segmenting RGBD images would be to simply use [5] after setting the edge weights to a weighted combination of differences in depth and color: $(1-\alpha)d_C(p_1,p_2)+\alpha d_D(p_1,p_2)$, where p_1 and p_2 are two pixels, d_C is the difference in color, d_D is the difference in depth, and α is a scalar weighting the relative importance between the two. We shall show in the experimental results in Figure 3 that this approach does not yield good results in practice, because there is no value of α that will work consistently well for a variety of images. In fact, in many cases no value of α works well even for the various regions of a single image. Instead we use the multistage approach described in Section 3.1.1.

RGBD optical flow: We compute optical flow between each consecutive pair of RGBD images. First, the depth values are converted to XYZ coordinates in 3D space by using

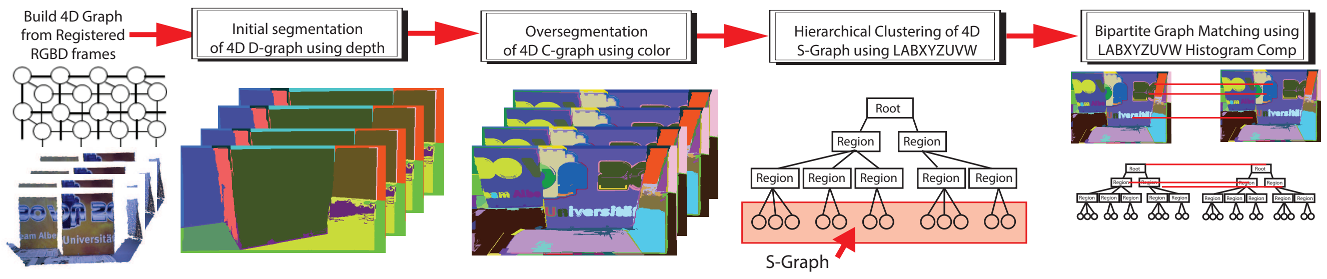
Build 4D Graph from Registered RGBD frames
Initial segmentation of 4D D-graph using depth
Oversegmentation of 4D C-graph using color
Hierarchical Clustering of 4D S-Graph using LABXYZUVW
Bipartite Graph Matching using LABXYZUVW Histogram Comp
Root
Region
S-Graph

Figure 2. Schematic of our method to segment streaming 3D point clouds.

the internal and external calibration parameters of the visible and IR cameras on the sensor to register the RGB and depth images. Then we compute dense optical flow between the two RGB images. Previous extensions of optical flow to 3D [2] assume a continuous spatiotemporal volume of intensity values on which differentiation is then performed. In our case, however, the depth values are already known, thus enabling a much simpler calculation. We compute 2D optical flow $(\Delta i, \Delta j)$ using a consecutive pair of RGB images based on Farneback's method [4]. Therefore, since the RGB and depth images are registered, for the value $D_{n-1}(i, j)$ in the depth map at location (i, j) at time n, there exists a corresponding depth value $D_n(i + \Delta i, j + \Delta j)$ in the next frame. The optical flow, w, along the z axis, is then simply the difference between the two depths:

$$w = D_n(i + \Delta i, j + \Delta j) - D_{n-1}(i, j). \tag{1}$$

We also project the motion of the (i, j) pixels into 3D space using the camera calibration parameters. Hence, the RGBD scene flow can be solved by extending Equation 1 and the calibration of the data and is defined as:

$$\begin{aligned}(u, v, w) =& ((i_n - i_{n-1})\frac{z_n}{F_x}, \\ & (j_n - j_{n-1})\frac{z_n}{F_y}, \\ & D_n(i + \Delta i, j + \Delta j) - D_{n-1}(i, j)).\end{aligned} \tag{2}$$

3.1.1 Segmentation using depth and color

Our approach relies on the following observation: If the scene consists only of convex objects, then every depth discontinuity corresponds to an object boundary. This observation was motivated by a study on primates [9] which concluded that depth and color perception are handled by separate channels in our nervous system that function quite distinctly. Obviously the world does not consist only of convex objects, but nevertheless we have found this observation to have great practical utility in everyday scenes. We exploit this observation by first segmenting based on depth alone, then based on color while preventing any merging to occur across depth discontinuities. We show in the experimental results that this approach yields more accurate segmentation than combining depth and color into a single graph.

We build a graph (called the D-graph) shown in Figure 2, in which each node corresponds to a toxel. Within each graph, nodes are connected to their 26-neighbors with edges whose weights are the absolute difference in the depth of the two toxels: $|D(x, y, z, t) - D(x', y', z', t')|$, where $D(x, y, z, t)$ is a function that yields the depth values from the spatiotemporal volume, and $(x', y', z', t') \in \mathcal{N}(x, y, z, t)$, where $\mathcal{N}(x, y, z, t)$ is the neighborhood of toxel (x, y, z, t). Edges are then constructed between each frame $D(x, y, z, t-1)$ and $D(x, y, z, t)$ by connecting toxels based on the computed optical flow. Toxel $(x, y, z, t-1)$ in D is connected with toxel $(x+u, y+v, z+w, t)$ in D with an edge whose weight is given by $|D(x, y, z, t-1) - D(x+u, y+v, z+w, t)|$. The internal difference of a region $Int(R)$ is set as the highest edge weight in the minimum spanning tree of region R, and regions are merged according to [5]. The constant that determines the size of depth segments is called k_{depth}.

After the graph-based algorithm produces a segmentation according to the depth, another graph (called the C-graph) is created with the same nodes as before shown in Figure 2. As before, edges are created for each of the 26-neighbors, and edges between frames are set according to the RGBD optical flow vectors to be $|C(x, y, z, t-1) - C(x+u, y+v, z+w, t)|$ where $C(x, y, z, t)$ is a function that returns the euclidean difference in color. We use CIE LAB color space to ensure perceptual uniformity of color differences. Invoking the observation above, the edge weights connecting toxels that belong to different regions after processing the D-graph are set to infinity in the C-graph to forbid merging across depth discontinuities. The constant that determines the size of color segments is called k_{color}. After running the graph-based algorithm on the C-graph, we have an over-segmentation consisting of super-toxel regions across the spatiotemporal hyperrectangle.

3.2. Hierarchical Processing

Once the over-segmentation has been performed, feature vectors are computed for each super-toxel region. For feature vectors, we use histograms of color, 3D position, and 3D optical flow, called LABXYZUVW histograms, using all the toxels in a region. For computational efficiency and storage requirements, rather than computing a single 9-dimensional histogram, we compute nine 1-dimensional histograms. Experimentally we find that 20 bins in each of the LAB and UVW features and 30 bins in each of the XYZ features is adequate to balance generalization with discriminatory power.

Using these feature vectors, a third graph (called the S-graph) is created to represent the regions as shown in Figure 2. Each node of the graph is a region computed by the super-toxel segmentation in Section 3.1.1 and edges are formed to join regions that are adjacent to each other (meaning that at least one toxel in one region is a 26-neighbor of at least one toxel in the other region). The weight of each edge is the difference between the two feature vectors, for which we use Equation 6. On this graph, instead of running the graph-based algorithm of [5], we use Kruskal's algorithm to compute the minimum spanning tree of the entire set of regions. The result of this algorithm is a dendrogram that captures a hierarchical representation of the merging in the form of a tree, in which the root node corresponds to the merging of the entire set.

By selecting a different threshold, a different cut in this minimum spanning tree can be found. To make the threshold parameter intuitive, we choose the percentage of regions to merge, which we denote by ζ. The value determines the coarseness of the produced results, with $\zeta = 0$ indicating that all the regions are retained from the previous step, whereas $\zeta = 100\%$ indicates that all the regions are merged into one. For values between these two extremes, the dendrogram is recursively traversed, accumulating all nodes until the desired percentage is obtained. Although the value ζ can be changed on-the-fly for any given image without incurring much computation to recompute the segments, we generally set $\zeta = 65\%$ for the entire sequence to avoid having to store all dendrograms for all frames.

3.3. Bipartite Graph Matching

After the dendrogram of the current pair of frames has been thresholded according to ζ, correspondence must be established between this thresholded dendrogram and that of the previous pair of frames. To achieve this correspondence, we perform bipartite graph matching between the two thresholded dendrograms using the stable marriage solution [6]. Figure 2 depicts the bipartite graph matching process, which considers the difference in region sizes, the change in location of the region centroid (after applying the 3D optical flow), and the difference between the LABXYZUVW histograms.

Data is treated as two lists from the tree cut. The histogram match from each region's perspective is found with the histogram SAD difference equation between Region R and Region S being:

$$\begin{aligned}\Delta H = \sum_{i=1}^{\text{NUMBINS}} &|\frac{R_l[i]}{R_N} - \frac{S_l[i]}{S_N}| + |\frac{R_a[i]}{R_N} - \frac{S_a[i]}{S_N}| \\ +|\frac{R_b[i]}{R_N} - \frac{S_b[i]}{S_N}| &+ |\frac{R_x[i]}{R_N} - \frac{S_x[i]}{S_N}| + |\frac{R_y[i]}{R_N} - \frac{S_y[i]}{S_N}| \\ +|\frac{R_z[i]}{R_N} - \frac{S_z[i]}{S_N}| &+ |\frac{R_u[i]}{R_N} - \frac{S_u[i]}{S_N}| + |\frac{R_v[i]}{R_N} - \frac{S_v[i]}{S_N}| \\ &+|\frac{R_w[i]}{R_N} - \frac{S_w[i]}{S_N}|\end{aligned} \tag{3}$$

where R_N is the number of toxels in Region R and S_N is the number of toxels in Region S.

For each histogram, we compute the distance the centroid has traveled over time and the size difference.

The difference the centroid has traveled is computed as the SAD of the center points normalized by the size:

$$\Delta d = \frac{|R_x - S_x| + |R_y - S_y| + |R_z - S_z|}{R_N} \tag{4}$$

and the size difference is computed as the absolute value of the difference:

$$\Delta N = |R_N - S_N| \tag{5}$$

Starting with the closest histogram, we merge the regions R and S iff they each consider each other the best choice using the weight as defined by Equation 6:

$$h = \beta \Delta H + \gamma \Delta d + \epsilon \Delta N \tag{6}$$

where β, γ, and ϵ are fixed constraints to prevent over-merging, found by computing statistics on multiple sequences and determining 3σ for each constraint.

4. Experimental Results

Our algorithm was evaluated with several 3D RGBD sequences, some from the NYU Dataset [18], some from the TUM Dataset [20], and some that we obtained. To measure the accuracy of the algorithm, we annotated some images by hand to provide ground truth as 2D object boundaries. To account for slight inaccuracies in the position error of a boundary point, we use the chamfer distance [3]. The boundary edges are extracted from the output segmentation images as well as of the ground truth images. For each pixel in the contour of the query image, we find the distance to the closest contour pixel in the ground-truth image (which is efficiently computed using the chamfer distance), and then we sum these distances and normalize by dividing by the number of pixels in the image. The error metric is shown in Equation 7.

$$E_{bound} = \frac{\sum_i \|NN^1(\text{Out}_i - \text{GroundTruth})\|}{\text{Width} * \text{Height}}, \tag{7}$$

Where NN^1 represents the closest point of the boundary output Out_i to the ground truth.

As mentioned earlier, we were unable to find a value for α that enables the traditional graph-based algorithm in [5] to produce robust, accurate results across all sequences. This is demonstrated in Figure 3, where the output for various values of α are shown, none of which clearly delineates the objects properly. The linear combination never properly segments the table, table legs, cup, magazine, and magazine colored features in Figure 3, whereas our over-segmentation does. In other words, the multistage segmentation combination of first depth and then color segmentation provides better results than any linear combination of color and depth features as the primate studies from [9] suggested.

To quantify these results, we evaluated the output of the graph-based algorithm of [5] for different values of α for all three sequences (shown in Figures 7-9 corresponding to S1-S3). As shown in Figure 4, better results are obtained when color is weighted higher than depth (α is small). Nevertheless, no value of α yields an acceptable segmentation. Also shown is the initial over-segmentation step of our method. Since our method is independent of α, these results are plotted as horizontal lines. Note that the segmentation error of our multistage segmentation algorithm is lower than the output from the linear combination approach for any value of α.

Even though our algorithm does not depend on α, it does initially depend on k_{depth} and k_{color}. The left of Figure 4 shows the error (measured by Equation 7 where k is the parameter from [5]) obtained from segmenting just based on depth as the parameter k_{depth} is changed. Note that on all three images the output gets worse as this value is increased due to under-segmentation. On the right is a plot of all three sequences for both approaches discussed in 3, where our algorithm includes only the initial over-segmentation step. With a low k_{depth}, these plots show the effect of changing the value of k_{color} for our algorithm, or similarly of k for [5]. For nearly all values our method achieves less error.

Figure 5 shows the error using the hierarchical method. On the left is a set of plots of all three sequences for both approaches, where we vary the hierarchical tree percentage cut. It can be seen that the hierarchy makes us invariant to the values of k_{depth} and k_{color} just as in [7]. On the right are plots of the error of our hierarchical step applied to the output of both our algorithm (initial over-segmentation step) and the algorithm of [5]. These plots answer two questions, namely, whether the hierarchical step improves results, and whether our initial over-segmentation step is needed (or whether the hierarchical step would work just as well if we simply used the algorithm of [5]). The answer to both questions is in the affirmative. The rightmost figure shows that our method also has a lower explained variation. Further proof that having depth is better than just color is shown in the comparison of RGBD images from the NYU Dataset [18] in Figure 6.

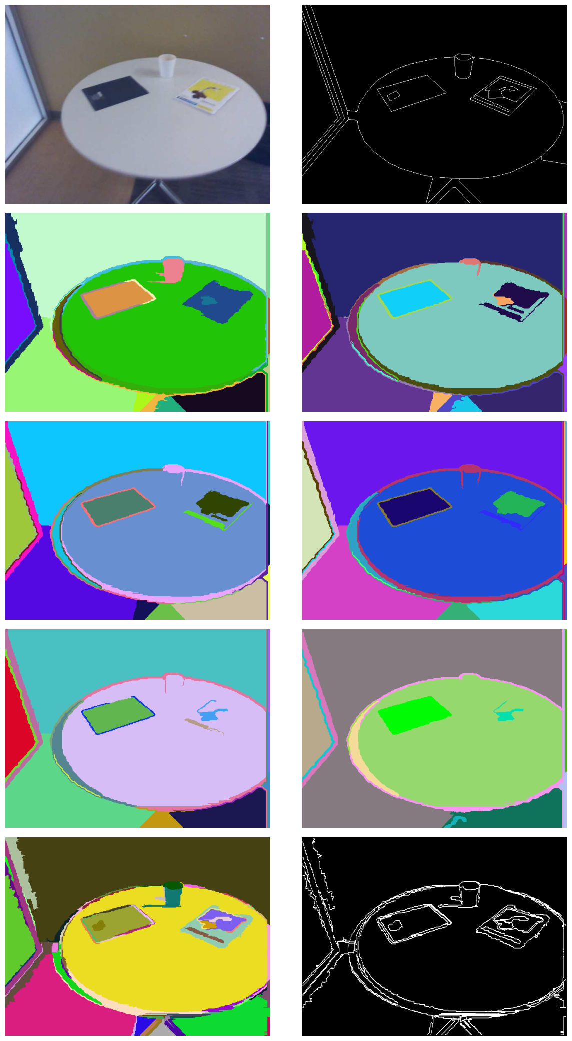
Figure 3. Top row: original RGB image (depth image not shown), and ground truth object boundaries. Middle three rows: The output of the graph-based algorithm using a linear combination of color and depth, with α values of 0.1, 0.2, 0.4, 0.6, 0.8, and 0.9; none produces a correct result. Bottom row: The output of the initial over-segmentation step of our algorithm (left), as well as the boundaries of this output (right).

Now that we have quantified our algorithm, we show segmentation results of the entire approach on three different 4D RGBD sequences compared to the state of the art video sequences of [24] and [7], shown in Figures 7, 8, and 9. Note that as people move around the scene, the camera moves, and brief partial occlusions occur, the segmentation results remain fairly consistent and the identity of regions are mostly maintained. It can be seen that our approach seems to always outperform [24] in both segmentation and temporal consistency. We maintain boundaries better and segment better than [7]. [7] also tends to merge segments to-

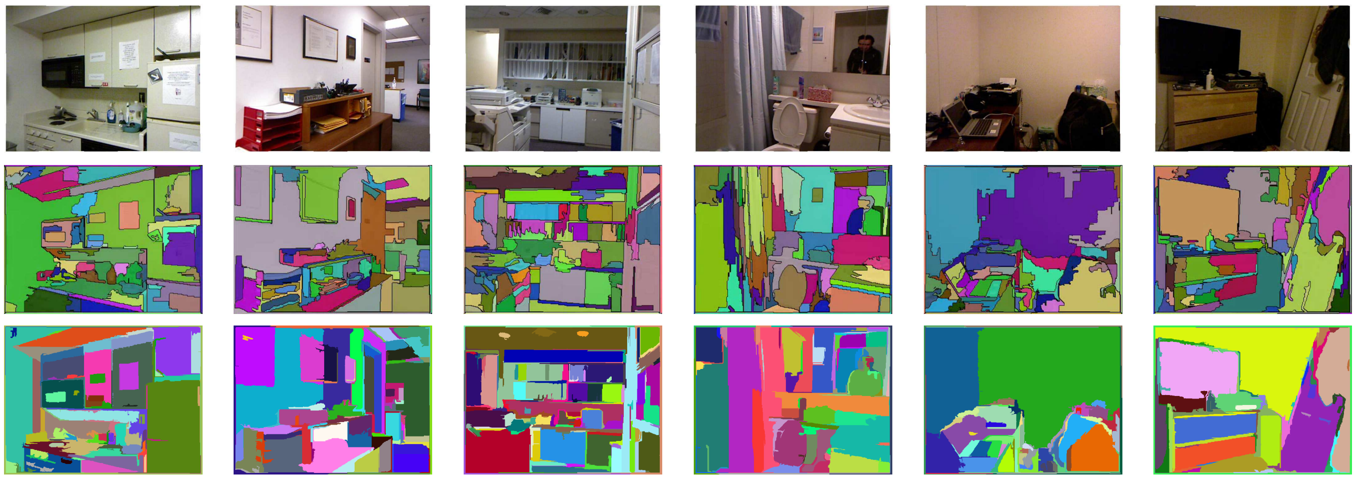
Figure 6. Segmentation results from different 3D RGBD scenes from the NYU data set[18]. The top row is the images, the 2nd row is the hierarchical result only using color with a hierarchical level of 0.7 described in [7], and the bottom row is our approach with a hierarchical level of 0.7.

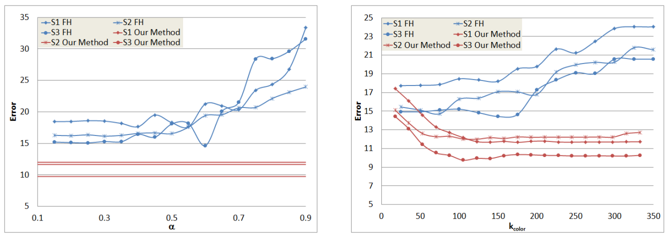

Figure 4. Left: The error based on the choice of α on the output of the linear combination algorithm on three sequences (blue lines), as well as the output of our hierarchical algorithm (red lines), which is not dependent on α Right: Effect of varying k_{color} on the output of our initial over-segmentation step, as well as the value of k on the output of [5] (with the best α possible.

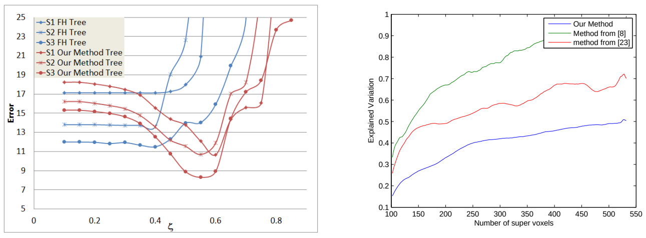

Figure 5. Left: Effect of varying ζ on the output of our hierarchical clustering applied to the output of either our initial over-segmentation step or that of [5] (with the best α and k possible). Right: The explained variation from [24] of our method, [24], and [7] on Sequence 2.

gether improperly such as the people merged into the background in Figures 8 and 9. Although [7] maintains temporal consistency better than our algorithm, it processes all frames at once as opposed to being stream-based and cannot run indefinitely, which is why last image in Sequence (C) is always N/A. Although there is a perceived error in boundaries, it is only due to the lack of overlap in the registration of the depth and color images inherent from the RGBD camera. Our algorithm runs at approximately 0.8 fps on 8 frame sequences of 640x480 RGB and depth images. [7] runs in 0.2 fps on downscaled 240x180 videos and [24] runs in 0.25 fps on downscaled 240x180 videos. See the project web page for videos.

5. Conclusion, Limitations, and Future Work

In this paper, we have presented an unsupervised hierarchical segmentation algorithm for RGBD videos. The approach exhibits accurate performance in terms of segments quality, temporal region identity coherence, and computational time. The algorithm uses a novel approach for combining the depth and color information to generate over-segmented regions in a sequence of frames. A hierarchical tree merges the resulting regions to a level defined by the user, and a bipartite matching algorithm is used to ensure temporal continuity. We showed that our method outperforms a linear combination of color and depth. We performed comparison against different graph segmentation combinations showing lower errors in terms of quality of the segmentation and thoroughly analyzing the effects of various parameters. There are occasional problems with noise from the depth image and maintaining temporal consistency. Future work will be aimed at improving results even further by incorporating boundary information, as well as incorporating higher-level information.

References

[1] A. Abramov, K. Pauwels, J. Papon, F. Worgotter, and B. Dellen. Depth-supported real-time video segmentation

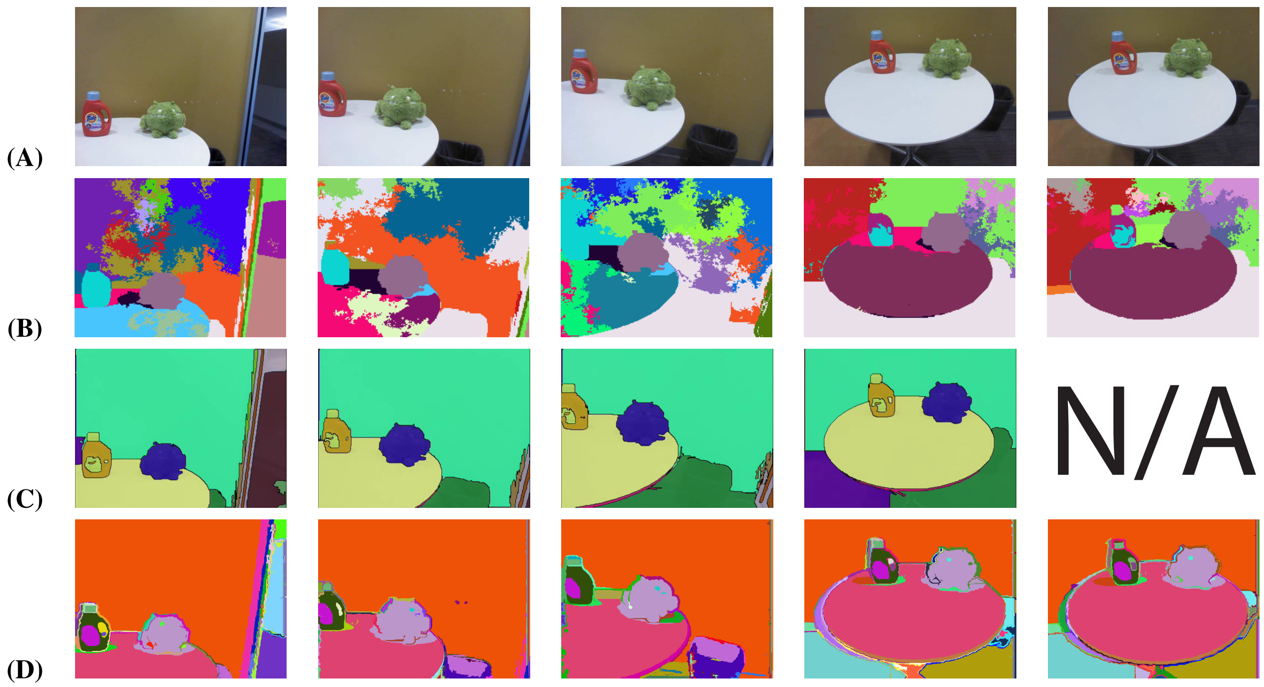

Figure 7. Sequence 1, (A) : RGB image, (B): Results from [24], (C): Results from [7] (N/A is used when the frame is beyond the length it can run), (D): Our Results

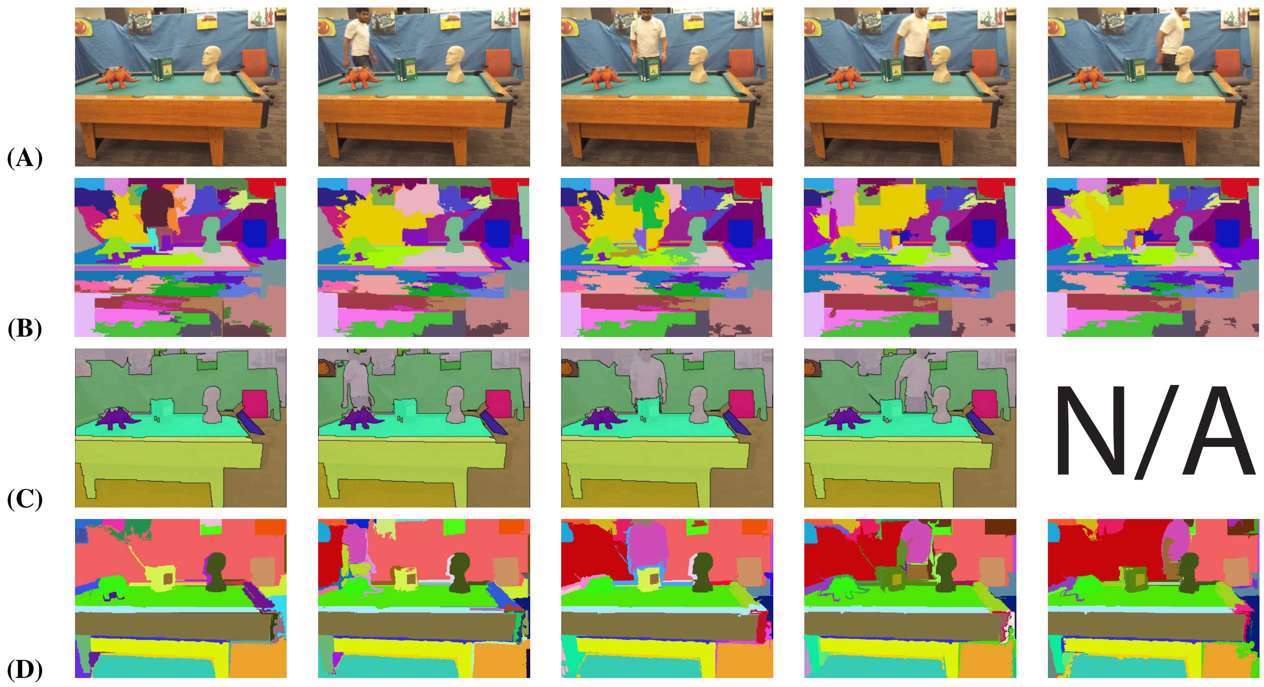

Figure 8. Sequence 2, (A) : RGB image, (B): Results from [24], (C): Results from [7] (N/A is used when the frame is beyond the length it can run), (D): Our Results

with the Kinect. In *IEEE Workshop on Applications of Computer Vision (WACV)*, pages 457–464. IEEE, 2012. 2

[2] J. Barron and N. A. Thacker. Tutorial: Computing 2d and 3d optical flow. Technical report, Imaging Science and Biomedical Engineering Division, Medical School, University of Manchester, 2005. 3

[3] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. *IJCAI*, pages 659–663, 1977. 4

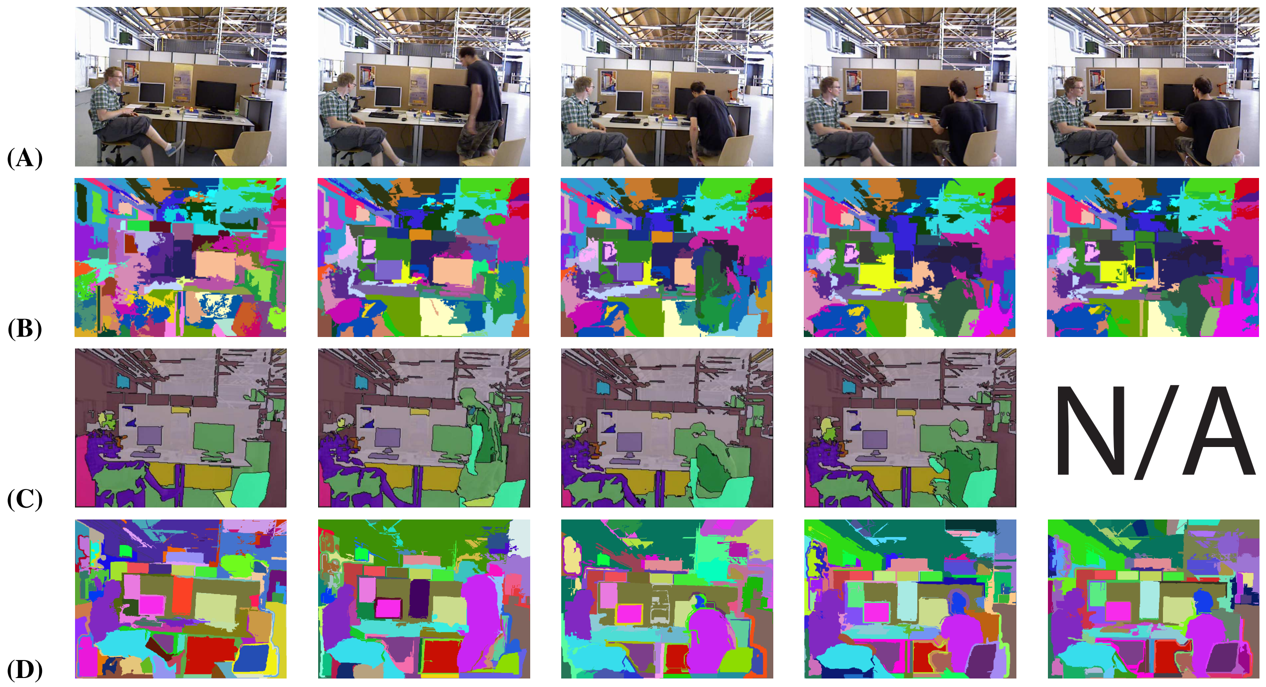

Figure 9. Sequence 3, (A) : RGB image, (B): Results from [24], (C): Results from [7] (N/A is used when the frame is beyond the length it can run), (D): Our Results

[4] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image Analysis*, pages 363–370. Springer, 2003. 3

[5] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 2(59):167–181, 2004. 2, 3, 4, 5, 6

[6] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962. 4

[7] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph-based video segmentation. In *CVPR*, 2010. 1, 2, 5, 6, 7, 8

[8] D. Holz and S. Behnke. Fast range image segmentation and smoothing using approximate surface reconstruction and region growing. In *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*, 2012. 2

[9] M. Livingstone and D. Hubel. Segregation of form, color, movement, and depth: anatomy, physiology, and perception. *Science*, 240(4853):740–749, 1988. 3, 5

[10] D. Munoz, J. A. Bagnell, N. Vandapel, and M. Hebert. Contextual classification with functional max-margin markov networks. In *CVPR*, 2009. 2

[11] S. Paris and F. Durand. A topological approach to hierarchical segmentation using mean shift. In *CVPR*, 2007. 1, 2

[12] V. A. Prisacariu and I. D. Reid. Pwp3d: Real-time segmentation and tracking of 3d objects. In *British Machine Vision Conference (BMVC)*, 2012. 2

[13] X. Ren, L. Bo, and D. Fox. Rgb-(d) scene labeling: Features and algorithms. In *CVPR*, 2012. 2

[14] A. Richtsfeld, T. Morwald, J. Prankl, J. Balzer, M. Zillich, and M. Vincze. Towards scene understanding object segmentation using rgbd-images. In *Computer Vision Winter Workshop (CVWW)*, 2012. 2

[15] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *ICRA*, 2011. Shanghai, China. 2

[16] E. Sharon, M. Galun, D. Sharon, R. Basri, and A. Brandt. Hierarchy and adaptivity in segmenting visual scenes. *Nature*, 442(7104):810–813, 2006. 2

[17] J. Shi and J. Malik. Normalized cuts and image segmentation. *TPAMI*, 22(22):888–905, 2000. 2

[18] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 2, 4, 5, 6

[19] J. Strom, A. Richardson, and E. Olson. Graph-based segmentation for colored 3d laser point clouds. In *IROS, 2010*, pages 2131–2136. IEEE, 2010. 2

[20] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IROS, 2012*, pages 573–580. IEEE, 2012. 2, 4

[21] A. Teichman and S. Thrun. Learning to segment and track in rgbd. In *The Tenth International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2012. 2

[22] D. Weikersdorfer, A. Schick, and D. Cremers. Depth-adaptive supervoxels for rgb-d video segmentation. In *ICIP*, pages 2708–2712, 2013. 2

[23] C. Xu and J. J. Corso. Evaluation of super-voxel methods for early video processing. In *CVPR*, 2012. 1, 2

[24] C. Xu, C. Xiong, and J. J. Corso. Streaming hierarchical video segmentation. In *ECCV*, 2012. 5, 6, 7, 8