

Segmentation of range images through the integration of different strategies

Mary Ellen Bock

Dept of Statistics, Purdue University
West Lafayette, IN 47907
Email: mbock@stat.purdue.edu

Concettina Guerra

Dip. Elettronica e Informatica
Università di Padova, Padova, Italy 35131
and Dept. Computer Sciences
Purdue University, West Lafayette, IN 47907
email: guerra@dei.unipd.it

Abstract

Segmentation of range images has long been considered an important and difficult problem and continues to attract the attention of researchers in computer vision. In this paper we consider the problem of segmenting range images into planar regions. The approach we present combines different strategies for grouping image elements to estimate the parameters of the planes that best represent the range data. The strategies differ not only in the way candidate planes are hypothesized but also in the objective function used to select the best plane among the potential candidates. The main contribution of the paper is in an effective integration of simple modules. Experimental results on a large set of real range images are presented.

1 Introduction

Segmentation of range images has long been considered an important and difficult problem and continues to attract the attention of researchers in computer vision. In this paper we consider the problem of segmenting range images into planar regions. The approach we present combines different strategies for grouping image elements to estimate the parameters of the planes that best represent the range data. The strategies differ not only in the way candidate planes are hypothesized but also in the objec-

tive function used to select the best plane among the potential candidates. All procedures are based on random sampling. The first two procedures *CoplanarLines* and *LinePoint* choose the best approximating planes on the basis of edge points only. The parameters of candidate planes are found by randomly selecting proper subsets of edge points: two coplanar lines, either parallel or not, for *CoplanarLines*, and a line and point not belonging to the line, for *LinePoint*. In both cases, the objective function for each hypothesized plane is the number of edge points that belong to the plane, i.e. are within ϵ distance from the plane, where ϵ is a small constant. The third procedure *AlternativeStrategy* generates a candidate plane that fits a small window of unlabeled range points and uses as objective function the total number of unlabeled range points that are within ϵ distance from the plane. An effective solution to the plane recovery problem is provided by a careful combination of the three procedures.

Much work has been done on the problem of extracting surface patches from range data [1, 2, 4, 8, 12]. Approaches to range image segmentation fall into two main categories: *edge-based* and *region-based* approaches. Region-based segmentation methods group range pixels into connected regions based on some homogeneity measure. Typically, they grow regions around *seed* points for which a planar fit gives a reliable estimate. In the edge-based segmentation approaches discontinuity points are first extracted from the range im-

age and then used to guide the segmentation process. Both approaches have their strength and weaknesses and need an extensive post-processing phase. Edge-based approaches tend to produce gaps in the boundaries of the regions while region-based approaches generate boundaries that are connected but generally distorted. A rigorous methodology for experimental comparison of range segmentation techniques was developed [7]. Within this framework, a total of seven planar segmentation strategies have been evaluated on a large set of real images using well defined performance metrics.

Robust estimation techniques have gained importance in image segmentation [3, 11, 13]. The objective of robust estimation is to reduce the influence of “outliers”, where an outlier is a point with a large residual from a model recovered. Random sampling schemes have the property of insensitivity to outliers. Thus they are especially convenient to handle data with many gross error in addition to measurement errors.

This paper is organized as follows. In section 2 we describe the framework for the segmentation problem. In section 3 we explain the three main procedures for plane recovery and their integration. In section 4 we introduce a number of parameters used by the program and, finally, in section 5 we present experiments conducted on a set of real range images.

2 General framework

The problem of image segmentation is a classical one and yet different definitions exist in the literature. Thus we begin by formally defining the problem we consider here. Let I be a range image and let (i, j) be a pixel of I and $f(i, j)$ its range value. The segmentation consists of partitioning the image I into regions R_1, R_2, \dots, R_n such that:

1. $\cup R_i \subseteq I, \quad i = 1, \dots, n$
2. $P(R_i) = true \quad \forall i$
3. $P(R_i \cup R_j) = false$ if R_i and R_j are adjacent
4. $R_i \cap R_j = \emptyset \quad \forall i \neq j$

where $P(R_i)$ is a predicate over the points of the set R_i . It defines the conformity of all points of R_i to the plane model:

$$P(R_i) = true \text{ if and only if } d(f(i, j), p) \leq \epsilon,$$

where d is a metric, p the plane model and ϵ a given constant.

An alternative definition adds to the above the following property:

$$R_i \text{ is connected } \forall i$$

One can view the segmentation process as the labeling of the image points according to the regions they belong to. Not including the above last property in our definition implies that regions may consist of unconnected parts that are assigned the same label (they are fragments of the same planar surface), while in the alternative definition separate surface patches have always distinct labels.

Property 1 is often replaced by:

$$1'. \cup R_i = I \quad i = 1, \dots, n$$

The reason for including property 1 instead of the 1' is that range image data are not perfect and there are often points that are cannot assigned any depth value and therefore remain unlabeled.

In the classical definition of image segmentation there is important variant of condition 3:

$$3'. P(R_i \cup R_j) = false \quad \forall i \neq j$$

For image segmentation using geometric parametric models the latter condition does not seem appropriate.

3 Plane recovery

The method we consider integrates in an effective way different strategies for plane recovery. There are mainly three procedures all based on random sampling. The first two procedures *CoplanarLines* and *LinePoint* choose the best approximating planes on the basis of edge points only. The parameters of candidate planes are found by randomly selecting proper subsets of edge points: two coplanar lines, either parallel or not, for *CoplanarLines*, and a line and point not belonging to the line, for *LinePoint*. In both cases, the objective function for each hypothesized plane is the number of edge points that belong to the plane, i.e. are within ϵ distance from the plane, where ϵ is a small constant set to 1 in our implementation.

The third procedure *AlternativeStrategy* generates a candidate plane that fits a small window of unlabeled range points and uses as objective function the total number of unlabeled range points that are within ϵ distance from the plane. This procedure tends to be computationally intensive due to the lengthy computation of the objective function. Thus, it is invoked only when the other two procedures fail to detect a good region.

At each iteration, to detect a new plane the three main procedures are invoked sequentially in a given order; the next one executed only when the previous ones do not detect a significant plane. The significance of a plane is expressed as the percentage of unlabeled range points that belong to the region. When this percentage is above a certain threshold S , the detected plane is accepted and all the image patches that belong to the plane are assigned a new label; otherwise, the plane is discarded. The threshold S is updated dynamically during the processing. It may happen that none of the procedures recovers a significant plane even though there are still many unlabeled points. This may be due to the fact that the threshold S is too high for this stage. Thus, the best plane among the ones detected by the three procedures is chosen and accepted as the new plane. Furthermore, the threshold S is updated and set to a value lower than the one corresponding to the significance of the new plane.

Once the parameters of the best plane are found at a given iteration, the plane is expanded over the entire image and all the fragments of the same surface in the image are labeled as belonging to the same plane. The fragments are generally due to occlusion but may also correspond to different faces of the same concave object or of different objects. Conflicts may arise because some points may be close to more than one plane. A typical case is a strip of points inside a region that accidentally happen to be within ϵ distance from a plane associated to some other region. A conflict at a range point is resolved in favor of the label with the highest number of occurrences in a small window centered at the point. There are cases, however, where a conflict cannot be resolved within a small local window and a more global strategy is needed.

In the following, we explain in more details the different modules of the method. A schematic diagram of the method is in figure 1.

Data structures

When a new plane is detected and accepted by one of the main procedures, its parameters are inserted into the list of detected planes. The edge points that are at distance less than δ from the plane are removed from the list of edge points; this latter list will be from now on referred to as the list the remaining edge points. δ is a constant larger ϵ (2.5 in our experiments) to account for errors in the mea-

surements as well as in the edge detection process. Furthermore, a new data structure is created and associated to the new plane. It stores a subset of the edge points belonging to the plane, that is the edge points with distance less than ϵ from the plane and classified as crease points by the edge detector. The list of edge points associated to a given plane is used in later stages of the algorithm to make hypotheses of new candidate planes having a line in common with this one. This data structure is updated during the processing each time a new plane is detected. All points of the list that are close to the new plane, are finally removed from the list. In fact, those points are at the intersection of two detected planes, the boundary of two image regions, and need no further consideration.

Another data structure is used in the algorithm and associated to each line extracted from the set of edge points. A line is defined by two edge points. In addition to that, all edge points in the given set that are within ϵ distance from the line are stored in a list. This list is used to give better estimates of the candidate planes through the line.

A plane from a seed in the range image

The procedure *AlternativeStrategy* randomly selects three uncolored range points (i, j) , $(i, j + 16)$, $(i + 14, j + 8)$ (the vertices of an almost equilateral triangle) and forms a plane with them. It considers the smallest rectangular window enclosing the three points and counts the number of uncolored range points in the window that are within ϵ distance from the plane. If such count is below a given threshold, the plane is discarded because it is considered not significant. The threshold is initially high so that almost all points in the window need to be covered by the plane and is relaxed towards the end of the processing when few range points are still uncolored. If the plane is accepted according to its local significance, we estimate the goodness of the fit over the entire range image by counting the total number of uncolored range points close to the plane.

The above steps are repeated several times and the plane that best fits the range points over the entire image is selected. The number of iterations depends on stage of the processing and is set experimentally.

A plane from a line and a point

The procedure *LinePoint* chooses a candidate plane defined by a line and a point outside the line. The line is the best fitting line for a collection of edge points. It is extracted from the set of remaining edge points, not associated to a previously detected plane. The line extraction in 3D is based on a simple approach that selects the optimal line among those that are defined by pairs of points in the dataset. It does so by determining for each line passing through a pair of edge points the number of other remaining edge points lying within the expected measurement error ϵ from that line; the best line is the one that corresponds to the maximum computed value. The number of pairs $\frac{n \times (n-1)}{2}$ is generally very large for any reasonable value of n . If all pairs of points have to be considered the overall $O(n^3)$ time complexity would be prohibitive. However, in practice a number of pairs much less than the above is acceptable. A relatively small subset of pairs of points randomly chosen can yield results that are within a given bound from the optimal.

The output of the line extraction is a pair of edge points belonging to the line. The pair of edge points for the line and a third point randomly selected in the set of remaining edges define a model plane. Several model planes are considered and the best one is chosen according to a figure of merit that takes into consideration all edge points currently assigned to other discovered planes as well as the remaining edge points. It is the number of edge points within ϵ distance from the plane.

A plane defined by two coplanar lines

The procedure *CoplanarLines* determines candidate planes defined by two coplanar lines. As described in the previous section, a line is the best fitting line for a set of edge points. Of the two lines, one at least is extracted from the set of remaining edge points and the second may also come from there or may belong to a previous plane.

Approximate coplanarity of the two lines is tested. To determine a plane associated with the two lines, three points are needed. The first two are the two edge points that determine the first line. The third point is one of the edge points in the epsilon neighborhood of the second line. Each of these third points determines a different candidate plane and the one finally chosen has the most number of edge points within epsilon of the two lines and also within epsilon of the candidate plane.

A plane defined by two coplanar lines detected in the scene does not necessarily correspond to an actual surface in the image. For instance, in a polyhedra with parallel faces, pairs of parallel lines not belonging to the same face generate planes that are not on the surface of the object. For a set of stairs in an image, the parallel lines that form the leading edge of two steps give an example. Thus, before making the final decision about the acceptance of a plane such cases have to be eliminated. This is done by drawing a line segment joining the two lines and checking that most of the points close to it lie in the candidate plane.

If two parallel lines are very close they are ignored. Close parallel lines are generally present at occlusion boundaries and do not form a region in the image. The number of lines considered is an input parameters.

Combining the three strategies

After having described the three main modules of our segmenter, we explain how they are combined to obtain a fast and effective overall method. At each iteration, to detect a new plane the three procedures are invoked one after the other with the next one executed only when the previous ones do not detect a significant plane.

The order of execution of the three main procedures CoplanarLines, LinePoint and AlternativeStrategy has been decided experimentally based on many runs with different permutations. The experiments were conducted on 40 range images acquired by an ABW structured light scanner.¹

Except for the first two planes, the procedures are now called in the following order: first CoplanarLines, second LinePoint and, finally, AlternativeStrategy. In the experiments on the ABW set of images using this order, the majority of the planes, approximately 75%, are recovered by CoplanarLines and only a few by LinePoint. The AlternativeStrategy is used mainly towards the end of the processing, to find small regions. In the determination of the first two planes, generally corresponding to the background and the ground, the procedure CoplanarLines is not included.

A crucial part of this method is the determination of the "significance" threshold S used to decide about the acceptance of a plane: if S is too small all

¹The ABW range images are available from <http://marathon.csee.usf.edu/range/segcomp/SegComp.html>.

planes returned as the best planes by the first procedure will be accepted and therefore the other two will almost never be executed; if, on the other hand, the threshold S is too high all three procedures will be executed at each iteration thus tremendously increasing the amount of work.

At this point we have to justify the use of three different modules and ask the question of whether the results obtained with a single module will be comparable with the results obtained by the integration of the three. Extensive experimentation has shown that each procedure would perform poorly in a way or another if used in isolation. If CoplanarLines is the only one used, then the first few planes are found quickly and effectively, however towards the end of the processing the procedure fails because pairs of lines cannot be detected reliably from the few edge points left in the list of remaining edge points.

AlternativeStrategy, on the other hand, is very effective both at the beginning of the processing to detect large regions (for instance, background or ground) as well as when most of the image points have already been labeled and only small patches are left. At intermediate stages, however the regions identified by AlternativeStrategy are not as sharp as if detected by other means and their boundaries tend to be somewhat distorted. In addition to that, this procedure is too time consuming to be used in all stages. The computation of the objective function for each candidate plane requires the determination of the distance of each unlabeled image point from the plane itself, which is much more work than determining the distance from the plane of the remaining edge points as done in the other two modules.

As for the last procedure LinePoint this is the fastest of the three; however the planes it recovers often fail to pass the significance test and therefore are rejected. This explains why the overall computation time when executing LinePoint first tends to be higher than executing CoplanarLines first. In the experiments, LinePoint has been successfully used for a relatively small percentage of all planes recovered.

Resolving Conflicts

After a plane is accepted, it is expanded over the image and all the fragments of the same surface are labeled. This is done by checking every range point to see if it is within the given approximation error

from the plane. If so, and if the pixel does not have a label already it is assigned the new label. If the point has a label already, the conflict is solved in favor of the label with the highest number of occurrences in a small window centered at the point.

It often occurs that a conflict cannot not be resolved because of an equal number of occurrences of two labels; in such cases the decision on how to label the pixels is postponed to the end of the processing after all planes have been detected. In the post-processing phase, a simple routine propagates labels from neighboring points to "unresolved" conflict points. Furthermore, the routine fills isolated points and removes small noisy regions.

4 Selection of parameters

We use several parameters in our program. A number of them deal with the random sampling paradigm and are used to specify the number of trials required to obtain a reasonable estimate of a parametric shape. We use random sampling for 1) line detection, 2) plane detection when the plane is constrained to contain a line; in such a case the plane is formed by either randomly selecting one point not on the same line or by selecting a coplanar line, and 3) for plane detection from a seed window.

The determination of the number of trials that are guaranteed to produce good results with high probability has been considered for the extraction of various parametric shapes and problems [5, 13]. In our system we derive reasonable values of these parameters from visual inspection and from comparisons with the ground truth images.

The procedure CoplanarLines uses the parameter nl as an upper bound for the number of lines extracted from the set of edge points and used to recover a plane. nl is set to 20 in our program. Each such line is the best line found in the set of edge points. After it is detected, the edge points closed to it are removed from the list and the line detection procedure is repeated. To determine the best line in a set of edge points we randomly select and examine n out of the $(n \times (n - 1))/2$ possible lines. A number of lines less than nl is used to hypothesize planes when a large fraction of the edge points have already been found close to some of the previous lines. This fraction fe is set to 0.7.

S is the significance value, i.e. a threshold on the percentage of unlabeled range points that must be

close to a plane to accept it. For the ABW images, a good choice for the initial value of S was found to be 0.15; in fact, in the ABW images a relatively large portion of the scene is generally covered by the ground and background. With this choice of S these two regions the first ones detected. S is decremented automatically in later stages.

The parameter T_{min} specifies the minimum size of a region; it is set to 200 range points. The parameter ws is used for the window size in the conflict resolution procedure. The value used for ws is 6×6 .

5 Experimental Results

We have conducted experiments on a set of 40 (512 x 512, 12-bit) images of polyhedral objects. This set together with ground truth images has been used in [7, 9] for an experimental comparison of seven range image segmentation algorithms. Of the 40 images, 10 have been used as training images to find the optimal values of the segmentation parameters. The outputs of our segmenter on the ABW images 15-29 are in figure 3².

We have compared these outputs with the ground truth images from [7] and the results are generally very good. However, a systematic comparison using the automatic procedure they developed is not possible since the ground truth images are made according to a different definition of the segmentation process (one that requires the detected regions to be connected). Only for some images of the set (typically, those that contain a single object) the ground truth images are valid for both definitions. In such cases the results of our algorithm have very high scores.

The program, written in C, has been implemented on a Sun Sparc5. The running times of the program on each of the 30 ABW images are reported in figure 2. Figure 2 displays the execution times (in seconds) of two different runs of the program. In the first run the order of execution of the three procedures is: CoplanarLines first, then LinePoint and finally AlternativeStrategy (light grey bar in the figure); in the other the order is: first LinePoint second CoplanarLines and finally AlternativeStrategy (dark grey bar). For the first variant of the program the average execution time is 315s while for the second is

352s.

References

- [1] P.J. Besl, R.C. Jain. "Segmentation through variable-order surface fitting" *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-10, n. 2, pp. 167-192, 1988.
- [2] M. E. Bock, C. Guerra. "A geometric approach to the segmentation of range images", *Proceedings of the Second International Conference on 3D-Digital Imaging and Modeling*, Ottawa, Canada, pp. 261-269, 1999.
- [3] K.L. Boyer, M.J. Mirza, G. Ganguly, "The robust sequential estimator: a general approach and its application to surface organization in range data," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 16, n. 10, pp. 987-1001, 1994.
- [4] T. Fan, G. Medioni, R. Nevatia, "Description of surfaces from range data using curvature properties," *Proc. Int. Conf. on Computer Vision and Pattern Recognition*, pp. 86-91, 1986.
- [5] M. A. Fischler, R.C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Computer Vision, Graphics and Image Processing*, vol. 24, pp. 381-395, n.6, 1981.
- [6] R. Hoffman, A.K. Jain, "Segmentation and classification of range images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-9, n.5, pp. 608-620, 1987.
- [7] A. Hoover, J. B. Gillina, X. Jiang, P.J. Flynn, H. Bunke, D. Goldgolf, K. Bowyer, D.W. Eggert, A. Fitzgibbon, R.B. Fischer, "An experimental comparison of range image segmentation algorithms," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-18, n. 7, 673-689, 1996.
- [8] R.C. Jain, A.K. Jain. *Analysis and interpretation of range images*. New-York:Springer-Verlag, 1990
- [9] X. Jiang, K. Bowyer, Y. Morioka, S. Hiura, K. sato, S. Inokuchi, M. Bock, C. Guerra, R.E. Loke, J.M.H. du Buf. "Some further results of experimental comparison of range image segmentation algorithms," *Proc. Int. Conf. on Pattern Recognition*, 1999.
- [10] X.Y. Jiang, H. Bunke, "Fast segmentation of range images into planar regions by scanline grouping," *Machine Vision Applications*, vol. 7, n.2, pp. 115-122, 1994.
- [11] K.M. Lee, P. Meer, Rae-Hong Park, "Robust Adaptive Segmentation of Range Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, n. 2, 1998.
- [12] A. Leonardis, A. Gupta, R. Bajcsy, "Segmentation of range images as the search for geometric parametric models," *Int. J. of Computer Vision*, vol. 14, n.3, pp. 253-277, 1995.
- [13] G. Roth, M. D. Levine, "Extracting geometric primitives," *Computer Vision, Graphics and Image Processing:Image Understanding*, vol. 58, n. 1, pp. 1-22, 1993

²Other results are available from <http://www.dei.unipd.it/guerra/Segmentation/Segmentation.html>

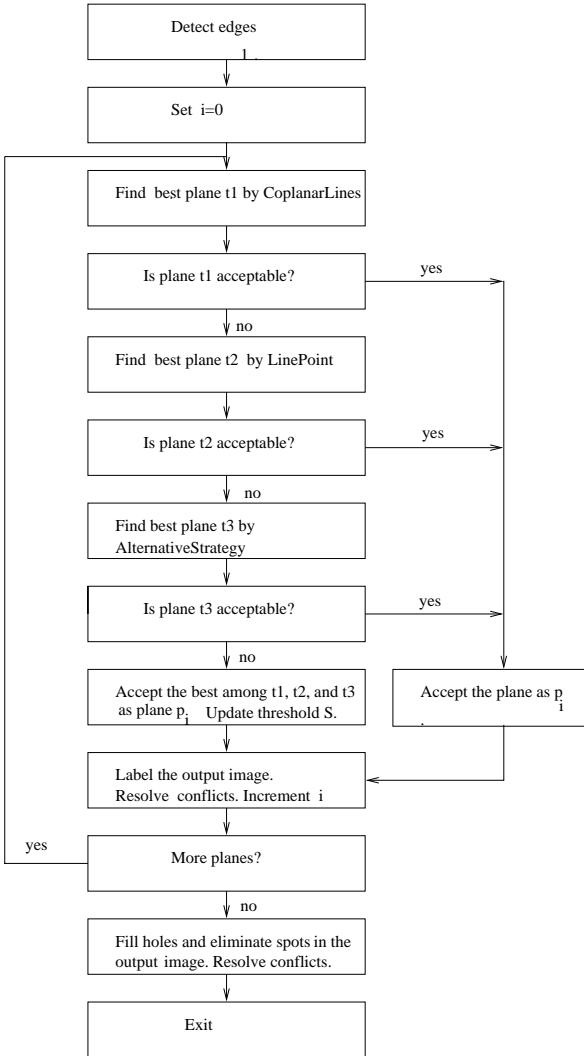


Figure 1: A schematic diagram of the segmenter

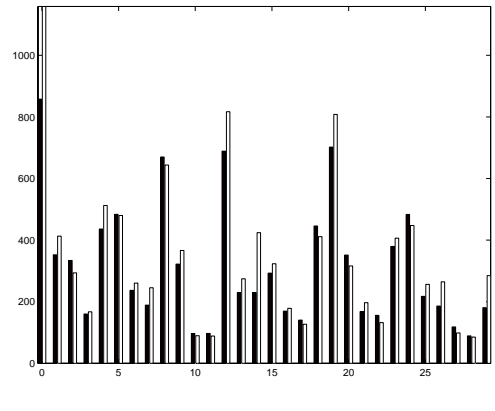


Figure 2: The execution times (in seconds) of two different runs of the program on each of the 30 ABW images

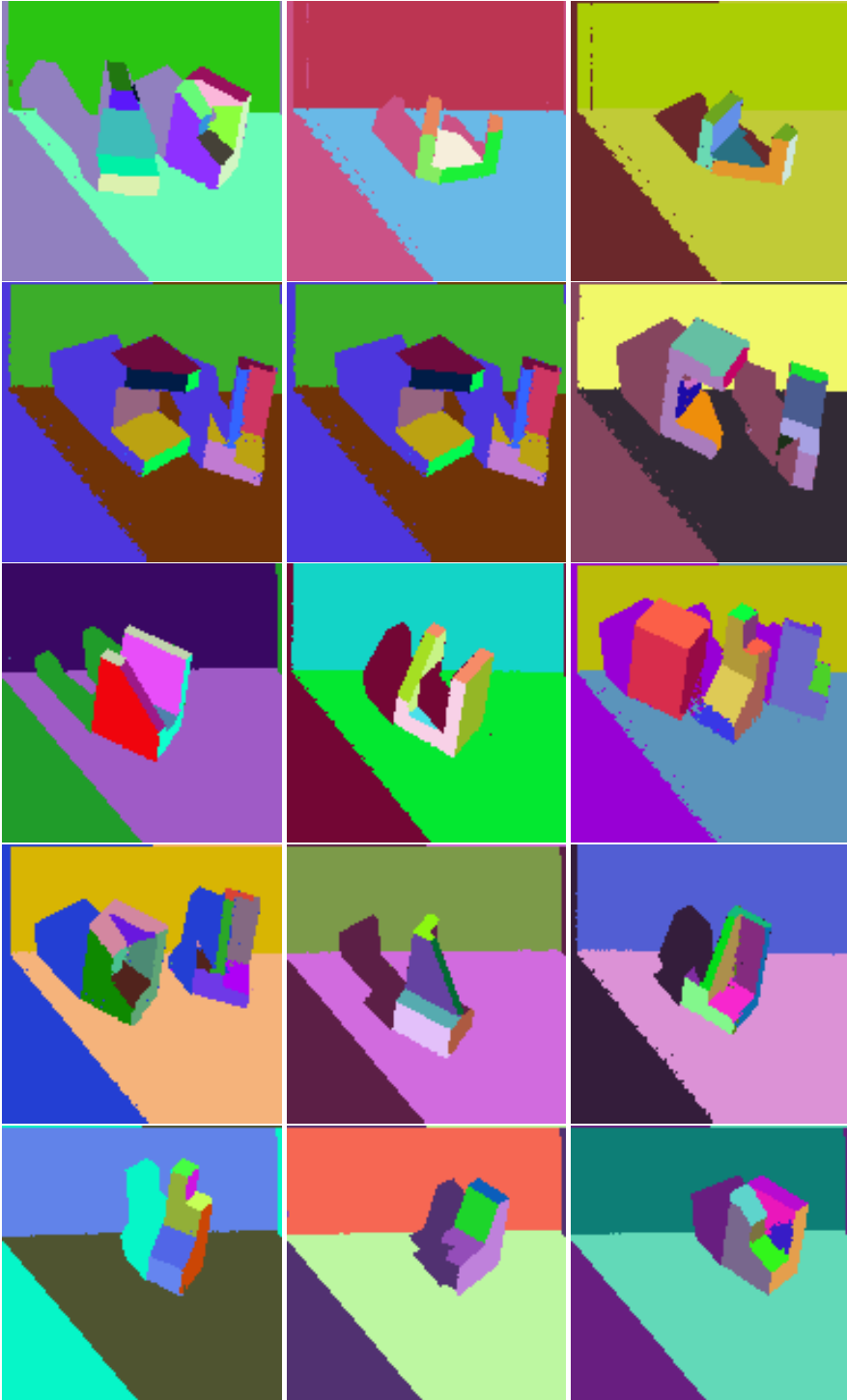


Figure 3: The output of our segmenter on the ABW images 15-29.