Simultaneous Image Segmentation and 3D Plane Fitting for RGB-D Sensors - An Iterative Framework

Li Guan Ting Yu Peter Tu Ser-Nam Lim
GE Global Research
1 Research Circle, Niskayuna, NY 12309, USA
{guan,yut,tu,limser}@ge.com

Abstract

In this paper, we segment RGB-D sensor (e.g. Microsoft Kinect camera) images into 3D planar surfaces. We initialize a set of plane equations based solely from the depth (point cloud) information. We then iteratively refine the pixel-to-plane assignment and plane equations. During this process, the number of planes are also reduced by merging adjacent local planes with similar orientations. For the pixel-to-plane assignment, we treat the image as a Markov Random Field (MRF), and solve the association problem using graph-based global energy minimization. We design the energy terms to encapsulate both appearance cues from the RGB (color) channels and shape cues from the D (depth) channel. Experiments show that the use of both appearance and geometry information significantly improves the segmentation quality, especially so at genuine plane edges and plane intersections. As a byproduct, the framework also automatically fills in missing depth information.

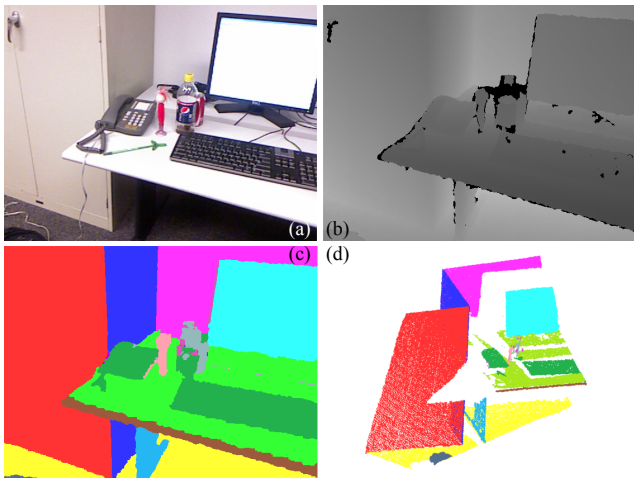

Figure 1. A typical indoor Scene and our plane recovery and segmentation. (a) original RGB frame from the Kinect camera. (b) depth frame aligned to the RGB view. Black regions indicate pixels missing depth values. (c) our final planar segmentation after the iterative segmentation converges. (d) a novel view of the converged 3D planes. Notice the true plane boundaries are preserved and pixels missing depth values are assigned with proper values.

1. Introduction

As shown in Fig. 1, an indoor scene usually consists of a collection of 3D planar surface patches, many of which are homogeneous, textureless regions, such as walls, table tops and floors. The knowledge of such planar structure would benefit numerous practical applications: for scene understanding and segmentation [22, 37], it not only provides semantic information of the general orientation of the scene, but also helps recognize typical objects in the scene; for Structure-from-Motion (SfM) [13] or multiview stereo, it helps plane matching to compensate for the lack of visual feature correspondences; for video denoising and compression, it helps restore colors based on the spatial consistency of the pixels in planar regions; and for LIDAR data registration, it helps Iterative Closest Point (ICP) [42] to more reliably align partially overlapping 3D point clouds.

Traditionally, for RGB images, only 2D regions can be segmented based on the color similarity; little better than that, for 3D point cloud data format, although meaningful 3D planes can be extracted with noise-resistant algorithms (*e.g.* RANSAC [11]), the planes tend to be unstable at intersections. They also tend to mistakenly merge visually distinctive and disconnected patches that are roughly parallel and have similar depth.

With the recent advances in the RGB-D sensor technology, such as the Microsoft Kinect, the simultaneously obtained RGB and depth images provide a chance to address the 3D plane segmentation problem beyond traditional methods. In this paper, we propose an iterative framework to simultaneously segment 3D planar patches from RGB-D frames and recover the corresponding plane parameters. After initialization, for each iteration, every plane equation is updated given the current pixels belonging to the plane;

978-1-4673-1612-5/12/$31.00 ©2012 IEEE

at the same time, we treat the image as a Markov Random Field (MRF), and solve the segmentation as a "pixel-to-plane" association problem using graph-based global energy minimization. We designed the energy terms so that they capture both the color/appearance and the depth cues. In addition, the iteration and global optimization will fill in the missing data (holes) with recovered planes.

2. Problem Overview and Related Work

Before the emergence of the RGB-D technology, image segmentation and point cloud (or depth image) plane fitting were considered to be two rarely overlapping research topics, both with quite a long history. For image segmentation, different algorithms have been well categorized in [16] into thresholding approaches [27, 9], contour-based approaches [31, 33], region-based approaches [1, 8], clustering approaches [24, 36, 10], and other optimization approaches, for example Bayesian framework [12] and neural networks [7]. As shown in later sections, the pixel-to-plane assignment (segmentation) procedure in our proposed framework uses thresholding, contour, region and optimization techniques to improve the output quality. One noticeable drawback with these unsupervised image segmentation techniques is that they lack the knowledge of the underlying scene geometry. Consequently, they tend to either over-segment due to the ambiguity between textures and real plane edges, or under-segment if two overlapping shapes have similar visual appearance.

For plane fitting to a point cloud, a group of algorithms focus on just fitting one dominant plane to a point cloud, and treat points outside of the plane as outliers, such as the RANSAC-like procedures used by the widely-used open source Point Cloud Library [34]. But a more difficult problem is to find all possible planes in a scene, and this is the main target of this paper. It is a very similar problem to color image segmentation, but the input now is the 3D point locations, which has been deployed in various approaches, such as depth discontinuity [2, 35], local surface normal similarity [30, 40] and scanline grouping [17, 26, 18]. Algorithm-wise, although all above approaches deploy different methods to handle minor sensor noise, such as consensus [34] and voting [5], and perform well when the dataset is of reasonable quality, they still fall into the category of deterministic methods. When the data is extremely noisy, probabilistic approaches [39] can be applied to explicitly model the underlying noise. System-wise, most of the reported methods use a "bottom-up" strategy to form planes from pixels. "top-down" methods [30, 40], on the other hand, start by assigning all the pixels to one group and fitting a single surface to it. They then keep subdividing the region until certain designed criteria are satisfied. As shown in later sections, our approach falls into the deterministic "bottom-up" category that uses both the depth discontinuity and surface normal similarity criteria. The main issue with the depth image/point cloud plane segmentation is that the only available information is the 3D locations. Although it reflects the general structure of the scene, it is unstable at the plane intersections, because points near a two-plane intersections satisfy both plane equations.

With both the color and geometry information available in one frame, the RGB-D cameras provide a great opportunity to combine the advantages of both image segmentation and plane fitting. Existing RGB-D algorithms focus on the high level usage or applications of this new technology, such as object recognition [4], 3D reconstruction [14], action recovery [25], detection and tracking [23] and foreground/background subtraction [3]. The only papers [15, 38] dealing with plane decomposition have not fully explored the RGB-D information. Although [15] claims to work with RGB-D cameras, it only uses the depth information for fast normal computation and dominant plane clustering. [38] discusses the same RGB-D image multi-plane segmentation problem. They begin with image segmentation, then uses N-cut to group super-pixels, and RANSAC to compute the final plane equations. The main drawback is that it highly depends on the initial segmentation, and does not have a refinement mechanism. As the authors pointed out, their method tends to under-segment and produces a single plane when two similar-colored planes overlap. It would not perform well at plane boundaries when two similar color planes are intersecting each other, such as at wall edges. Both the above drawbacks are because their segmentation is a one-pass process and does not have a refinement stage. Its final result also does not treat pixels with missing depth information. Recently [37] assigns every pixel in an RGB-D frame to a high-level object class, such as a bookshelf, sofa *etc.* It is a supervised segmentation method in the sense that it needs to learn the priors for a fixed number of classes.

In this paper, we segment an RGB-D frame into planar regions, which is one level lower than semantic segmentation. Although we also resort to graph-labeling optimization, our approach is unsupervised, namely the pixel-to-label(plane) prior only depends on the geometric property between the pixel and plane. Our proposal has three main advantages. First, we view the plane equation recovery and pixel-to-plane association (segmentation) as two closely related problems. Unlike existing approaches [15, 38], we realize that the improvement of the one can help improve the other. We thus propose to simultaneously refine the two steps in an integrated iterative framework. This is extremely effective when dealing with ambiguous pixels near plane edges/intersections. Second, we try to model the RGB-D information using consistent energy terms, and solve the segmentation using a global energy minimization framework.

The energy representation allows us to flexibly explore a number of RGB-D cues. Furthermore, the global energy minimization can be used to fill pixels with missing depth information. Unlike the bilateral filter type [29] of filling used in [37] based only on local color information, our filling scheme uses computed plane equations, which are much more constrained and accurate. Finally, we choose to initially over-segment the image and merge them into final planes as the refinement iterations proceeds. This makes our approach suitable for general situations, since it waives the requirement that the number of image regions has to be a known prior as in many earlier works [41, 28, 37].

The following sections are organized as follows: Section 3 formulates the problem and introduces our solution; Section 4 evaluates the algorithm with experiments and analyses; Section 5 discusses a few possible applications, extensions and concludes the paper.

3. General Framework

The complete framework is as shown in Algo. 1. Given I_{RGB} the RGB image, I_D the depth image and K the camera intrinsics as input, the algorithm outputs a set of 3D planes and their plane-point associations. Later in this section, we explain in detail how to initialize the planes and how to refine the results iteratively using both color and depth cues in the energy optimization scheme. Also we explain the criteria to reduce redundant planes and perform plane merging.

Algorithm 1: RGB-D segmentation.

Input: I_{RGB}, I_D and K.
Result: 3D planes and their plane-point associations.
Initialize pixel groups and the corresponding plane equations of the grouping (Sec. 3.2, Algo. 2);
while *pixel-to-plane assignment* ***NOT*** *stable* **do**
  Compute new pixel-to-plane assignment via Fast-PD energy optimization (Sec. 3.3);
  Re-compute planes given new pixel assignment;
  Merge planes if necessary (Sec. 3.4);
end

3.1. RGB-D View Alignment

Depending on the sensor modality, an RGB-D sensor's raw color image and depth image are not necessarily aligned, such as the Kinect sensor. Although some SDKs provide function calls to align the RGB and depth views of a Kinect sensor in a black box manner, in general, we can align such non-aligned RGB-D cameras as follows. Suppose we know the intrinsics of both the color and depth camera lenses (K_{RGB} and K_D respectively) and the relative location and orientation between the two lenses (R and t respectively), we are able to find the corresponding RGB values of any depth image pixel according to Eq. (1).

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim K_{RGB} \, [R \mid t] \left(d \cdot K_D^{-1} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \right), \qquad (1)$$

where $[x, y, 1]'$ and $[x', y', 1]'$ are the corresponding image homogeneous coordinates of the raw RGB and depth image pixels, and d is the depth value from the depth image.

From now on, we assume that we are dealing with RGB images I_{RGB} and depth images I_D that are aligned. The sensor intrinsic matrix K is also known. For a 3D surface location $[X, Y, Z]'$ in the camera's coordinate system, its color and depth information are stored at location $[x, y]'$ of I_{RGB} and I_D respectively. The values of $[x, y]'$ can be computed with the expression as in Eq. (2). Note that Z is in fact the depth value stored at location $[x, y]'$ in I_D. In other words, our input is a set of 3D point clouds with color information.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim K \begin{bmatrix} X/Z \\ Y/Z \\ 1 \end{bmatrix}. \qquad (2)$$

The coefficients of the plane equation in homogeneous form $[a, b, c, 1]'$, such that $[X, Y, Z, 1] \cdot [a, b, c, 1]' = 0$, is the plane normal vector. It can be solved as the null space of the N matrix consists of the 3D points as in Eq. (3). More specifically, it corresponds to the rightmost singular vector of the Singular Value Decomposition (SVD) of N. When $n = 3$, the solution is exact. When $n > 3$, it is equivalent to a least square solution.

$$N = \begin{bmatrix} X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \\ \vdots & \vdots & \vdots \\ X_n & Y_n & Z_n \end{bmatrix}, \quad n \geqslant 3, \qquad (3)$$

3.2. Region-growing Plane Initialization

To initialize for the iterative segmentation, we compute a set of planes directly from the depth image (the point cloud). Our method is a bottom-up "region-growing" method: we start from the top-left corner of the image location as a seed point, check its neighboring pixels to determine whether the 3D points corresponding to these pixels make a reasonable plane. If so, we keep growing this region until no more neighboring pixels can be explained by the same plane. We then move on to the next un-visited pixel as a new seed until all pixels in the RGB-D image have been explored.

Since the iterative refinement is designed to merge existing planes rather than generating new planes or splitting existing planes into sub-planes, one critical requirement is that the number of seed planes should be larger than the actual number of planes in the scene. Normally, this requirement is satisfied, because our initialization usually consists

of more than 100 plane patches, as shown in the leftmost picture of Fig. 5.

In practice, we apply a few empirical criteria for robust pixel grouping. First of all, only neighboring pixels whose corresponding 3D locations are close enough with the existing plane equation are added for plane growing. This decision is guarded by a threshold t_{point}. To some degree, it avoids the case of using points of two distinctively different depth values to compute the one plane equation. It also avoids using pixels without depth information. Second, we compute every 3D point's local normal beforehand using its local neighborhood region. We vary the neighborhood size so that it is proportional to the distance of the seed point to the camera. This complies with the intuition that the larger the distance, the noisier the depth measurement is, and the more samples we need to draw to compute a reliable normal. When we check whether to add a certain 3D point for plane growing, the angle between the point's local normal and the existing plane normal should be smaller than a threshold t_{angle}. Finally, when no more 3D points in the neighborhood can be added for the existing plane, we recompute the plane equation. We keep the new plane only when its distance to every contributing point is shorter than t_{plane}. The complete algorithm is as follows in Algo. 2.

Algorithm 2: Plane segmentation initialization

Input: 3D points computed from K and I_D.
Result: 3D planes and their plane-point associations.
compute local normal for every 3D point;
while *not all 3D points visited* **do**
  pick a seed from the un-visited points;
  compute a plane with the seed and its neighboring points via **SVD**;
  while *neighboring points satisfy* t_{point} *and* t_{angle} **do**
    add the point;
    re-estimate the plane via **SVD**;
    if *new plane satisfies* t_{plane} **then**
      update the plane equation and point-plane association list;
    end
  end
end

Note that although we have the aforementioned criteria to enforce the segmentation quality, it is still possible to mis-classify pixels in the initialization. It is exactly why we need a framework to refine the plane equations and re-examine the pixel-to-plane association. Note also that we only use geometrical information for initialization. Color information will be used in the iterative refinement.

Also to simplify the computation, especially to reduce the number of SVD operations, which is computationally expensive, we only perform plane re-estimation when there are no more neighboring points that can be added by examining t_{point} and t_{angle}. When the list of points to calculate the plane equation is larger than 5000, we only randomly choose 5000 points from the point list for SVD. Progressive SVD solutions such as [32] may also be applicable here with less computation overhead.

3.3. Graph Optimization

We now iteratively refine the plane equations and pixel-to-plane associations using all information available from an RGB-D frame. The reason why we only check color (consistency) information after the initialization is that colors for a whole plane are not necessarily the same, due to different textures or unbalanced lighting on the plane. During the iterative refinement, however, we look at the pixel scale. It is reasonable to assume that neighboring pixels have similar colors. As shown in Algo. 1, every iteration consists of two steps: pixel-to-plane re-association and plane equation re-estimation.

We do not want to determine the plane association for a single pixel at a time. For example, pixels near plane intersections agree with both plane equations. Not to mention that sensor noise could potentially worsen the labeling quality. Therefore, we have to achieve a global optimal solution or at least within a guaranteed bound. We treat the image grid of the RGB-D frame as a Markov Random Field (MRF), and the goal is to assign every pixel to the most reasonable (maximal posterior probability) plane equation "label". As a common strategy [6, 21], instead of directly solving the probability maximization, we equivalently solve the energy minimization as a discrete graph labeling problem in the form of Eq. (4).

$$E_{MRF} = \sum_{X} \sum_{Y \in \mathcal{N}(X)} E_{XY}(l_X, l_Y) + \sum_{X} E_X(l_X). \quad (4)$$

The label set $\mathcal{L} = \{l^1, \cdots, l^n, \phi\}$ represents the n planes plus a null label ϕ denoting pixels that do not belong to any plane. Set $\mathcal{N}(X)$ is the 4-connected neighborhood system of point X in the 2D graph. $E_X(l_X)$ is the **unary energy**. It only concerns with the likelihood of a pixel's corresponding 3D point belonging to a plane. $E_{XY}(l_X, l_Y)$ is the **binary energy**. It represents the graph neighborhood similarity or in other words concerns with the prior probability (cliques).

We measure the unary energy from two aspects: "Point-Plane Distance" d_p^2 and "Plane Normal Similarity" d_a^2. The formulation is shown in Eq. (5)-(7),

$$E_X = d_p^2 \cdot d_a^2. \quad (5)$$

Point-Plane Distance It is simply the square of the distance d_p^2 of the 3D point to the candidate plane.

$$d_p^2 = \frac{(aX_0 + bY_0 + cZ_0 + 1)^2}{a^2 + b^2 + c^2}, \tag{6}$$

where $aX + bY + cZ + 1 = 0$ is the plane equation and $[X_0, Y_0, Z_0]'$ is a 3D point's location.

Plane Normal Similarity The intuition is that for two points belonging to a same plane patch, not only the 3D points' locations should physically be very close, but their local normals should also be similar to that of the assigned plane. The more the point's local normal and the plane normal agree with each other, the smaller the energy d_a^2 would be, as in Eq. (7), with a range $d_a^2 \in [0, 1]$.

$$d_a^2 = \left(2 - \frac{1}{\pi/2} \cdot \arccos\left(\frac{n_{point} \cdot n_{plane}}{|n_{point}| \cdot |n_{plane}|}\right)\right)^2. \tag{7}$$

Although we computed 3D points' local normal in the initialization, we revisit the normal computation with the help of RGB color information. The improvement in the local normal computation here also illustrates the reason why deploying different sensing modalities such as color and depth is an effective way to achieve higher vision algorithm quality. The main difference is that instead of using a fixed-size pixel neighborhood for normal computation, we throw away the pixels within the neighborhood that are color-wise inconsistent with the seed pixel. The color consistency measurement is the same as in the *Color Similarity* discussed later. Fig. 2 shows the visual difference of the two normal computations. The visual-aided version has shaper normals especially at the plane edges.

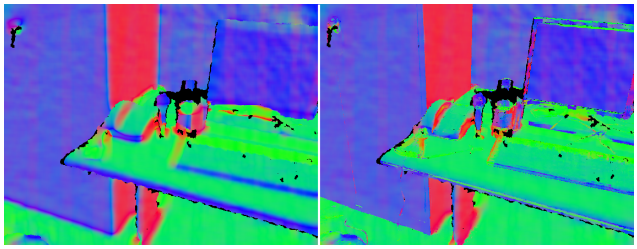

Figure 2. Normal maps, with R, G, and B channels representing the three orthogonal components of a 3D normal vector. Left: local normals computed from geometric information; Right: visual-guided local normals, which are sharper and more consistent, *e.g.* around plane boundaries near the floor and monitor.

We measure the binary energy from four aspects: "Color Similarity" d_c^2, "Mutual Local Distance" d_{pp}^2, "Mutual Local Normal Angle" d_{aa}^2 as well as "Visual Edgeness"d_e^2. The formulation is shown in Eq. (8)-(12),

$$E_{XY} = w_1 d_c^2 + w_2 d_{pp}^2 + w_3 d_{aa}^2 + w_4 d_e^2, \tag{8}$$

where w_1 to w_4 are the weighing parameters of the system.

Color Similarity d_c^2 is simply the sum of square distance in the RGB color space:

$$d_c^2 = (R_i - R_j)^2 + (G_i - G_j)^2 + (R_i - R_j)^2. \tag{9}$$

Mutual Local Distance d_{pp}^2 is the sum of square distance from point X to Y's local plane and Y to X's:

$$\begin{aligned} d_{pp}^2 =& \frac{(a_Y X_X + b_Y Y_X + c_Y Z_X + 1)^2}{a_Y^2 + b_Y^2 + c_Y^2} \\ &+ \frac{(a_X X_Y + b_X Y_Y + c_X Z_Y + 1)^2}{a_X^2 + b_X^2 + c_X^2}, \end{aligned} \tag{10}$$

where $[a_X, b_X, c_X]'$ is the local normal at point X: $[X_X, Y_X, Z_X]'$ and $[a_Y, b_Y, c_Y]'$ is the local normal at point Y: $[X_Y, Y_Y, Z_Y]'$. The local normals are computed with the help of RGB color information as shown in Fig. 2.

Mutual Local Normal Angle d_{aa}^2's form is similar to Eq. (7). It is the angle between the local normals of X's and Y's:

$$d_{aa}^2 = \left(2 - \frac{1}{\pi/2} \cdot \arccos\left(\frac{n_X \cdot n_Y}{|n_X| \cdot |n_Y|}\right)\right)^2. \tag{11}$$

Visual Edgeness The intuition is that comparing to normal pixels, a pixel that is visually an edge is more likely to be at the intersection between two physical planes. The edgeness term d_e^2 is measured as the square of local color image gradient magnitude. Fig. 3 shows the edgeness map of the RGBD image in Fig 1 (a)&(b).

$$d_e^2 = \left(\frac{\partial I_{RGB}}{\partial x}\right)^2 + \left(\frac{\partial I_{RGB}}{\partial y}\right)^2 + \left(\frac{\partial I_D}{\partial x}\right)^2 + \left(\frac{\partial I_D}{\partial y}\right)^2. \tag{12}$$

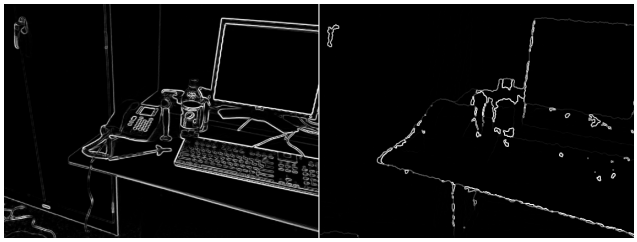

Figure 3. Edgeness maps computed from Eq.(12). Left: edgeness of the color image in Fig.1 (a); Right: edgeness of the depth image in Fig.1 (b). Notice some edges are only detectable in one of the two modalities, *e.g.* the desk leg and the vertical cabinet edges.

It is easy to see that any one of the above empirical criterion does not guarantee a 100% correct labeling, but the combination of all different cues robustifies our re-association. We choose to solve such discrete graph labeling problems using the Fast-PD approach [21] which builds

upon principles drawn from the duality theory of linear programming in order to efficiently derive almost optimal solutions for a very wide class of NP-hard MRFs [20, 6]. One of its main advantages is that it handles cost functions with arbitrary pair-wise potentials, lifting the *sub-modularity* constraint of previous approaches [19]. This gives us the freedom to develop relatively sophisticated energy constraining terms, as introduced above, from all the information we have from the RGB-D channels.

3.4. Plane Re-computation and Merging

From the initialization in Sec. 3.2, we deliberately oversegment the planes. Now in the refinement stage, in every iteration, after the point-to-plane assignment is computed, the plan is to recompute the plane equation parameters and merge the planes, as in Algo. 1. Once the new plane equation is computed given the re-assigned points using SVD, we try to check any two planes to see if they can be merged. The merging criterion is as follows: Suppose we have two point lists P_1 and P_2. We only examine such lists if they are connected, namely at least one point in P_1 and one from P_2 are neighbors with each other. Now suppose P_1 is associated to plane $a_1X + b_1Y + c_1Z + 1 = 0$, whose average to-plane distance is $\bar{d}_1$, and P_2 is associated to plane $a_2X + b_2Y + c_2Z + 1 = 0$, whose average to-plane distance is $\bar{d}_2$. Denote the angle between the two plane normals as α, the SVD solution of the merged plane computed from both P_1 and P_2 as $a_0X + b_0Y + c_0Z + 1 = 0$, and the average to-plane distance from both P_1 and P_2 to $a_0X + b_0Y + c_0Z + 1 = 0$ as $\bar{d}_0$. We merge the two planes if all terms of Eq. (13) satisfy.

$$\begin{cases} \alpha \leqslant \pi/18 \\ \bar{d}_0 \leqslant 1.2\bar{d}_1 \\ \bar{d}_0 \leqslant 1.2\bar{d}_2 \end{cases} \tag{13}$$

4. Experiments and Evaluation

We capture sequences of real indoor scenes under normal lighting condition with a hand-held moving Microsoft Kinect camera and perform plane segmentation to all frames with one fixed set of parameter values. The segmentation results are shown in Fig. 4, in which we compare our results with two benchmark algorithms: the purely color-based segmentation algorithm (mean-shift clustering) and a variation of [5], which is a 3D point-cloud plane fitting based on polar-coordinate Hough transform. Since the distance from a plane to the world space origin is very sensitive to the plane normal errors, unlike [5] which finds the plane normal and plane-origin distance in one voting space, we first vote for the plane normal, and refine the distance by fixing the normal. The dominant planes (voting space peaks) are depicted in the fourth column in Fig. 4.

Qualitatively, from this comparison we notice the following. First, color-only image segmentation may "flood" into neighboring patches, if the colors are similar, *e.g.* two vertical wall regions in Fig. 4(2c)&(4c). It would also oversegment a region if the colors are different, such as the photos located on the wall in Fig. 4(3c). Second, point-cloud-based plane fitting does not have sufficient evidence to extrapolate and fill in missing data such as the occluding boundary. Whereas, our hole filling guided by the plane equations and color similarity constraints produces reasonable results in areas like the ceiling light in Fig. 4(2) and the monitor rims in Fig. Fig. 4(1)&(4). Third, in order to maintain robustness against the noise or outliers, pure plane fitting is incapable of disambiguating two planes whose orientation and distance only have subtle difference, such as the keyboard and desk top in Fig. 4(1d). But thanks to the color information from the RGB image, we are able to recover such slightly different planes. Finally, although our planar world assumption helps constrain the segmentation in many ways, and is in general suitable for indoor environment, with non-planar shapes such as the wires in Fig. 4(2e), the planar approximation may look unnatural, but still, it is an optimal linear approximation to the surface.

Quantitatively, for the plane fitting quality, the point clouds to the recovered planes are within a mean distance of 0.470cm and standard deviation of 0.954. In comparison, the baseline method in Fig. 4(d), namely the modified Hough transform method recovers planes only from the point cloud with a mean distance of 0.992cm and standard deviation of 3.664. This shows that we recover a more accurate scene structure than just 3D point cloud plane fitting.

For the segmentation quality, we evaluate both *accuracy* and *fragmentation* in a single measure Q_{ratio}. For the ground truth, we manually segment a frame into planar regions. For every ground truth segment G^p, we measure $Q^p = |M^*|$, where M^* is the intersection G^p and the image segment produced by method M that has the largest overlap with G^p. $|\cdot|$ denotes the area of a segment. $Q_{ratio} = \frac{\sum_{i=1}^{n} |Q^i|}{\sum_{i=1}^{n} |G^i|}$, where n is the number of ground truth segment. Ideally, if we have a perfect segmentation, namely the segments all align with the manual result, Q_{ratio} should be 1, otherwise it is between zero and one. The more accurate the segment is to the ground truth, the closer Q_{ratio} value is to 1. The fewer pieces a ground truth segment is broken into, the closer Q_{ratio} value is to 1. We measure $Q_{ratio}^{mean\ shift} = 0.7653$ and $Q_{ratio}^{our\ method} = 0.9156$. This again indicates that our method produces better segmentation than color-only segmentation.

Convergence and speed. For all 80 frames, the algorithm converges within 30 iterations. Fig. 5 shows a number of intermediate labeling for the frame in Fig. 1(a)&(b), which

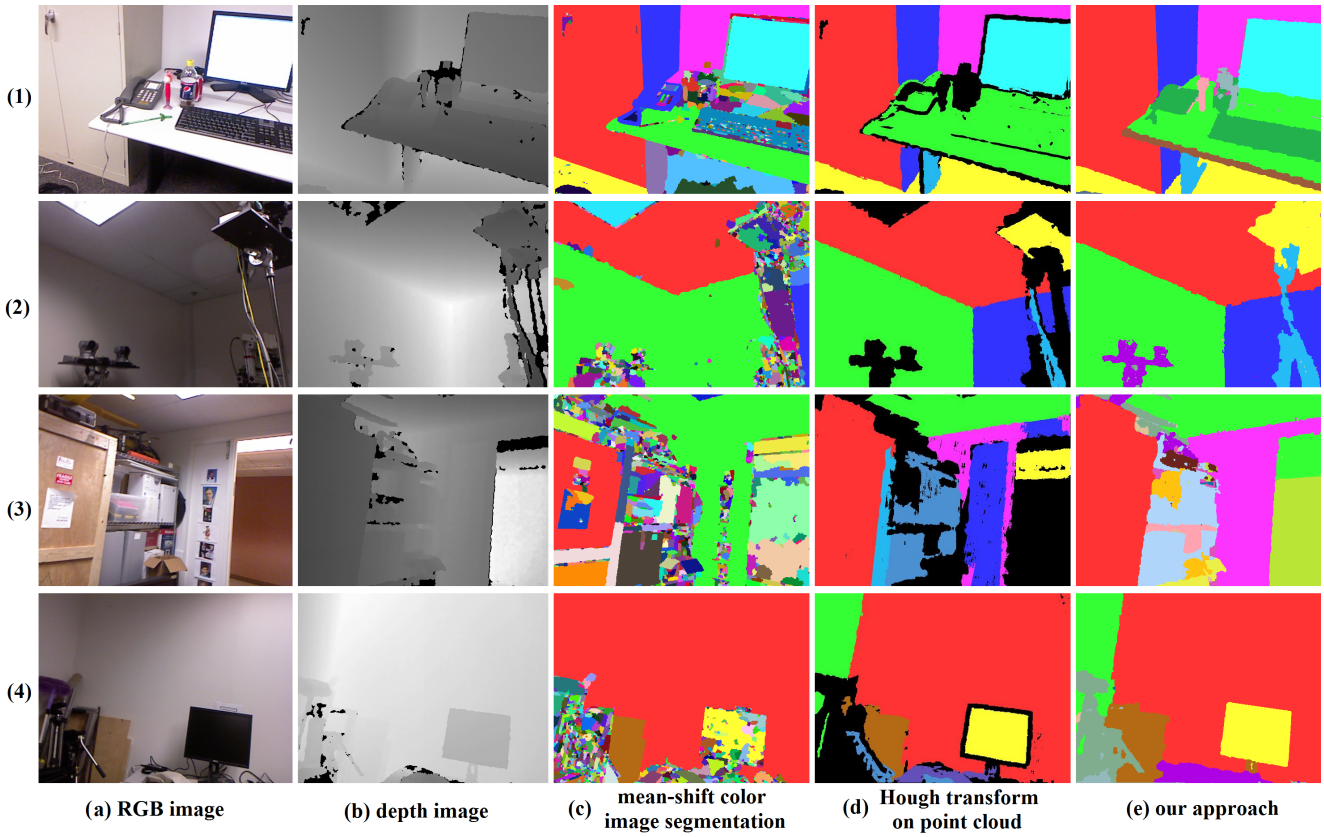

Figure 4. Algorithm result comparison. Each row shows the inputs and results of an RGB-D frame. Our approach which uses both color and geometry information clearly out-performs the two baseline approaches: the mean-shift color image segmentation [10] and a modification of [5] that uses Hough transform for 3D point cloud plane fitting.

converges in 17 iterations. We observe that the initial planes with more than one hundred labels quickly reduce to a few labels just after one iteration, due to the labels' spatial consistency enforced by the pair-wise graph energy. Planes are further merged and refined until they converge. In terms of computation speed, we tested our single-thread C++ version on 3.2GHz a quad-core machine with 3.26GB memory. It takes about 20 sec. to initialize the plane patches due to region-growing with SVD re-computation, which can be accelerated similar to [32] and with a GPU parallelized seed testing. Every main iteration takes about 2 seconds.

5. Discussion

In this paper, we introduced an iterative framework for simultaneous 3D plane segmentation and refinement from RGB-D images. This framework is mostly suitable for an indoor scene consisting of planar surfaces. It combines both the color and geometry information available in an RGB-D frame and does not need to know the number of existing planes in advance. The geometry-guided hole filling can be beneficial to many applications.

Future work includes algorithm speed up and exploring more complicated shapes beyond 3D planes. Although planar surfaces are the main structures in an indoor scene, we can extend to work with more sophisticated structures, as long as the analytical representations are known, such as cylinders or spheres. The proposed iterative framework is exactly the same to segment a scene and simultaneously recover the structure equations.

References

[1] J. Adams and L. Bischof. Seeded region growing. *PAMI*, 1994. 2

[2] B. Bhanu, S. Lee, C. Ho, and T. Henderson. Range data processing: representation of surfaces by edges. *ICPR*, 1994. 2

[3] A. Bleiweiss and M. Werman. Robust real fusing time-of-flight depth and color for real-time segmentation and tracking. *DAGM Workshop on Dynamic 3D Imaging*, 2009. 2

[4] L. Bo, K. Lai, X. Ren, and D. Fox. Object recognition with hierarchical kernel descriptors. *CVPR*, 2011. 2

[5] D. Borrmann, J. Elseberg, K. Lingemann, and A. Nuchter. The 3D hough transform for plane detection in point clouds - a review and a new accumulator design. *3D Research*, 2011. 2, 6, 7

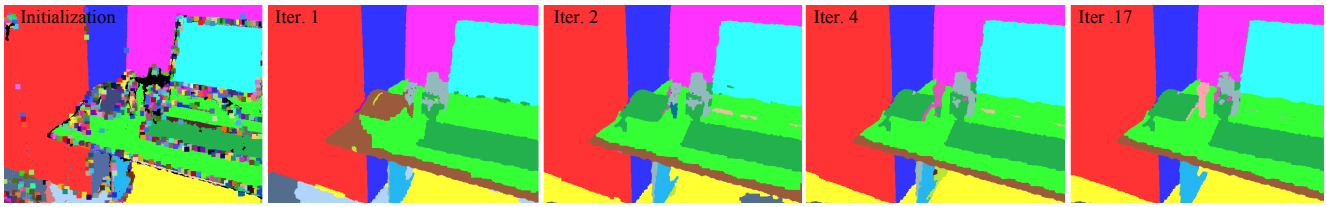

Figure 5. Algorithm convergence analysis. For initialization, more than 100 plane patched are formed, the majority of which are at the real plane boundaries which are merged in later steps.

[6] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 2001. 4, 6

[7] N. Campbell, B. Thomas, and T. Troscianko. Automatic segmentation and classification of outdoor images using neural networks. *International Journal on Neural Systems*, 1997. 2

[8] Y. Chang and X. Li. Adaptive image region growing. *IEEE Transaction on Image Processing*, 1994. 2

[9] M. Cheriet, J. Said, and C. Suen. A recursive thresholding technique for image segmentation. *IEEE Transaction on Image Processing*, 1998. 2

[10] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 2002. 2, 7

[11] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Com. of ACM*, 1981. 1

[12] S. German and D. German. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *PAMI*, 1984. 2

[13] R. Hartley and A. Zisserman. Multiple view geometry in computer vision. *Cambridge University Press*, 2003. 1

[14] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using depth cameras for dense 3d modeling of indoor environments. *RGB-D workshop, RSS*, 2010. 2

[15] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke. Real-time plane segmentation using RGB-D cameras. *RoboCup Symposium*, 2011. 2

[16] B. Jeon, Y. Yung, and K. Hong. Image segmentation by unsupervised sparse clustering. *WACV/MOTION*, 2005. 2

[17] X. Y. Jiang, U. Meier, and H. Bunke. Fast range image segmentation using high-level segmentation primitives. *WAVI*, 1996. 2

[18] I. Khalifa, M. Moussa, and M. Kamel. Range image segmentation using local approximation of scanlines with application to CAD model acquisition. *Machine Vision Applications*, 2003. 2

[19] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *PAMI*, 2004. 6

[20] N. Komodakis and G. Tziritas. Approximate labeling via graph-cuts based on linear programming. *PAMI*, 2007. 6

[21] N. Komodakis, G. Tziritas, and N. Paragios. Fast, approximately optimal solutions for single and dynamic MRFs. *CVPR*, 2007. 4, 5

[22] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view RGB-D object dataset. *ICRA*, 2011. 1

[23] M. Luber, L. Spinello, and K. Arras. Learning to detect and track people in rgbd data. *RGB-D workshop, Robotics Science and Systems conference*, 2011. 2

[24] L. Lucchese and S. K. Mitra. Unsupervised segmentation of color images based on k-means clustering in the chromaticity plane. *IEEE Workshop on Content-based Access of Images and Video Libraries*, 1999. 2

[25] D. Ly, A. Saxena, and H. Lipson. Pose estimation from a single depth image for arbitrary kinematic skeletons. *RGB-D workshop, RSS*, 2011. 2

[26] E. Natonek. Fast range image segmentation for servicing robots. *ICRA*, 1998. 2

[27] R. B. Ohlander. Analysis of natural sciences. *PhD Thesis, Carnegie-Mellon University, Pittsburgh*, 1975. 2

[28] M. Omran, A. Engelbrecht, and A. Salman. Dynamic clustering using particle swarm optimization with application in image segmentation. *ICCI*, 2005. 3

[29] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. *ECCV*, 2006. 3

[30] B. Parvin and G. Medioni. Segmentation of range images into planar surfaces by split and merge. *CVPR*, 1986. 2

[31] W. A. Perkins. Area segmentation of images using edge points. *PAMI*, 1980. 2

[32] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak. Fast plane detection and polygonalization in noisy 3D range images. *IROS*, 2008. 4, 7

[33] J. M. Prager. Extracting and labeling boundary segments in natural sciences. *PAMI*, 1980. 2

[34] R. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). *ICRA*, 2011. 2

[35] A. Sappa and M. Devy. Fast range image segmentation by an edge detection strategy. *3DIM*, 2001. 2

[36] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 2000. 2

[37] N. Silberman and R. Fergus. Indoor scene segmentation using a structured light sensor. *ICCV Workshop on 3D Representation and Recognition*, 2011. 1, 2, 3

[38] C. Taylor and A. Cowley. Segmentation and analysis of rgb-d data. *RGB-D Workshop, Robotics Science and Systems conference*, 2011. 2

[39] J. Weingarten, G. Gruener, and A. Dorf. Probabilistic plane fitting in 3D and an application to robotic mapping. *ICRA*, 2004. 2

[40] R. Xiang and R. Wang. Range image segmentation based on split-merge clustering. *ICPR*, 2004. 2

[41] Y. Zhang. Image engineering and related publications. *International Journal of Image and Graphics*, 2002. 3

[42] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *IJCV*, 1994. 1