INDOOR SCENE 3D MODELING WITH SINGLE IMAGE

A Thesis

Presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

CHUANMAO FAN

Ye Duan, Thesis Supervisor

DECEMBER 2015

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

INDOOR SCENE 3D MODELING WITH SINGLE IMAGE

presented by Chuanmao Fan

a candidate for the degree of master of science,

and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Ye Duan

Dr. Yunxin Zhao

Dr. Gang Yao

ACKNOWLEDGEMENTS

I would like to thank my wife, Wei Wei, and daughter, Elaine Fan, for their patience and tolerance during the times I had to work even though they needed extra care and help. Without the support of my family, I could not have finished this study.

I also would like to thank Xu Wang, who helped me complete the project work. His help during the holidays, weekends and in the evenings when others were having happy times with their friends and family is truly appreciated. I would also like to thank Truce for sharing his time and work in image segmentation. I would like to thank all who has helped me through the project work.

I would like to thank my advisor Dr. Ye Duan for his guidance in the process of the project. I learned fundamental professional knowledge in the early studies which allowed me to come into the field of computer vision and computer graphics. I obtained patient instruction from him during project research work, which I greatly appreciated. I have changed study topics or orientation several times during the study. Without the support and help of Dr. Duan, I would not have been able to overcome such obstacles. It has been an honor to have him as my adviser and to have his instruction.

Finally, I would like to thank my committee members for reviewing my thesis. I really appreciate Dr. Yunxin Zhao and Dr. Gang Yao for spending their precious time and providing so many valuable comments.

Thank you all.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

3D modeling is a fundamental and very important research area in computer vision and computer graphics. One specific category of this research field is indoor scene 3D modeling. Many efforts have been devoted to its development, but this particular type of modeling is far from mature. Some researchers have focused on single-view reconstruction which reconstructs a 3D model from a single-view 2D indoor image. This is based on the Manhattan world assumption, which states that structure edges are usually parallel with the X, Y, Z axis of the Cartesian coordinate system defined in a scene. Parallel lines, when projected to a 2D image, are straight lines that converge to a vanishing point. Single-view reconstruction uses these constraints to do 3D modeling from a 2D image only. However, this is not an easy task due to the lack of depth information in the 2D image. With the development and maturity of 3D imaging methods such as stereo vision, structured light triangulation, laser strip triangulation, etc., devices that gives 2D images associated with depth information, which form the so called RGBD image, are becoming more popular. Processing of RGB color images and depth images can be combined to ease the 3D modeling of indoor scenes. Two methods combining 2D and 3D modeling are developed in this thesis for comparison. One is region growing segmentation, and second is RANSAC planar segmentation in 3D directly. Results are compared, and 3D modeling is illustrated. 3D modeling is composed of plane labeling, automatic floor, wall, and boundary point detection, wall domain partitions using automatically detected wall, and wall boundary points in 2D image, 3D

modeling by extruding from obtained boundary points from floor plane etc. Tests were conducted to verify the method.

CHAPTER 1. INTRODUCTION

1.1 **Background**

3D modeling has recently emerged as a very hot research topic. Partially due to the growth of 3D imaging techniques such as stereovision, structured light imaging, and scanning laser triangulation, 3D devices can provide high quality 3D point cloud images directly or images associated with depth information. Depth information is lost in 2D imaging due to the projective principle of photometry. For decades, researchers have made efforts to recover 3D information from 2D photos through single-view reconstruction and modeling [1, 2]. This is a very challenging task. Only a few methods have proven successful. Most of the successful images leverage vanishing points detected directly from the 2D image [3, 4]. Besides, 2D indoor scene images and sub-urban scenes further make use of a strong geometrical constraint wherein the under-shot scenes fulfill the Manhattan world assumption [5, 6]. Furthermore, the additional depth information or direct 3D information provided in modern 3D imaging and photography has made the task of leveraging vantage points from 2D images for 3D transformation easier. The more information we have, the easier the task is.

Neverova et.al. used both the Manhattan world assumption and single low-quality RGBD images to produce a global 2½D model to exploit both color and depth information at the same time to fully represent an indoor scene from a single Kinect RGB-D image using geometry estimation. Rectified depth image and 3D scene normal are given as results [7]. Silberman et al. made use the of RGBD image to assist with indoor scene segmentation [8]. Their proposal developed a conditional random filed (CRF) energy function to measure the cost of a latent label y over each pixel, from which label y takes a set of values $\{1, \ldots C\}$ as the segmented regional

values. This model is trained by local prior. Further, Siberman et al. [8] presented an approach to detect the surfaces and objects and explore the support relations of the indoor scene. Their method can successfully parse complicated support relations of messy indoor scenes. However, their method made use of prior training, which was difficult to use. Taylor and Cowley [9] presented an approach to parsing the indoor scene using RGBD image. Their method detects the floor plan with an optimization labeling through dynamic programming. They segment the color image first based on the edges extraction, which is used as a prior in the second step to search for the planar regions. Planar regions are obtained by a Random Sample Consensus (RANSAC) routine which creates a segmentation that can sort through complex environments and find relevant groupings which cuts iterations and allows the user to more quickly discover structures of interest [10, 35]. Floor is identified subsequently, and the dominant surfaces' normal are obtained, which are then used to estimate the layout.

1.2 **Aim of the study and method.**

In this study, we reconstruct the 3D indoor scene using an RGBD image. The whole procedure is divided into two parts. The first part is the preprocessing procedure, which is illustrated in Figure 1.

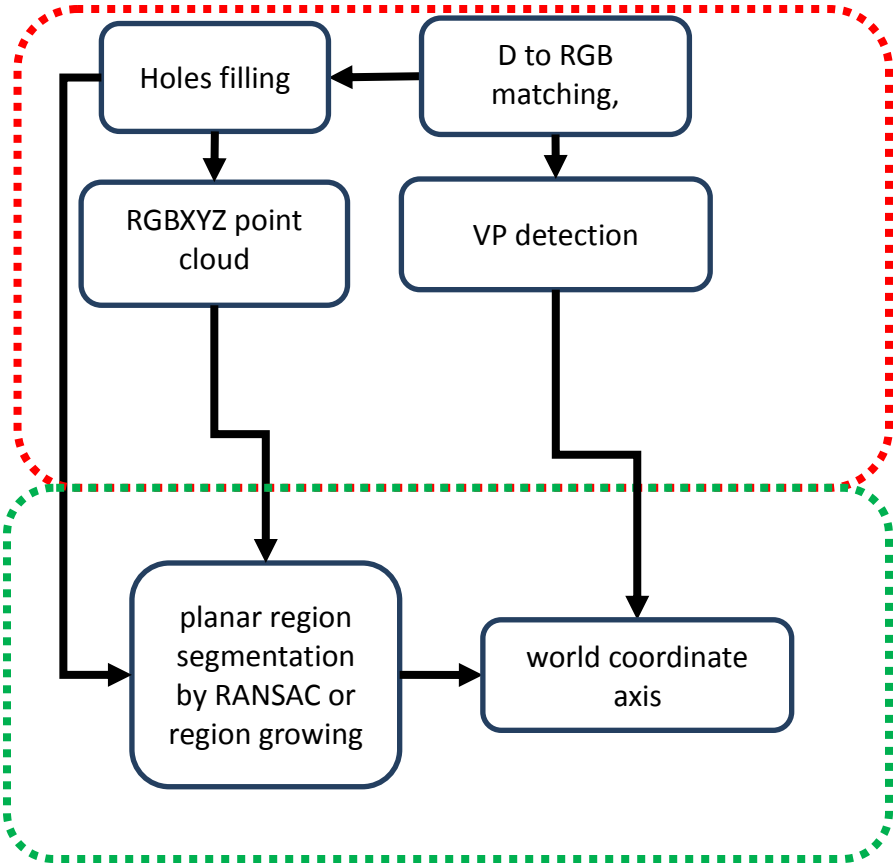


Figure 1. 1 Preprocessing for 3D modeling of indoor scene

Figure 1 refers to the RGB, which is the color image from a color imager. The depth image D is generated with a structured light projector plus a monochrome camera pair. In our study, the device providing these images was Kinect (Microsoft Inc.). Details about RGBD are given in Section 1.3. In the preprocessing, D to RGB matching was achieved to transform the depth image to an RGB color image, which is required because color image and depth image must be processed together to model an indoor scene and the two images are not registered. The D imager provides the depth image located at a position different from the color RGB imager. Thus, the first step was to register these two images. Due to the pinhole camera modeling of these two imagers, the two images were matched by a 3D world translation and rotation between two imagers. After the images were registered, they were then cropped to leave only an

overlapped part. Holes may form where there is no structured light reflectance information or because of un-decodable structured light patterns or dead zones of the two imagers. Holes affect the modeling, so it is filled before continuing the process. A bilateral filtering process associated with color imaging is adapted to fill holes. Together with the calibration matrix of the color imager and depth information registration, the RGB+D image can be used to generate color 3D point clouds.

VP detection is vanishing points detection from the 2D color image. We adopted Lezama's method [11] whereby three vanishing points are usually obtained. VP detection had two functions in our study: 1) vanishing points can be used to get the world coordinate unit axis of the Manhattan world scene. The world coordinate axis can help to recognize the wall and ground planes' normal alignment, which can then be used to detect wall and ground orientation. The 3D planes of floor and wall are detected by plane fitting on planar region 3D points with the help of the planar regional detection methods RANSAC or the region growing method. 2) The y vanishing point which produces the world unit axis when aligned with ground plane normal can be used to find the wall plane's left and right extension margin in the 2D image. Planar region segmentation is a very important step in 3D modeling. It gives the initial planes that may become the real plane of the 3D model.

Two methods are presented for planar region segmentation, i.e., planar region growing and RANSAC plane fitting. Planar region growing starts from several seeds given by either the user or from initial segmentation or every pixel in the image. Image pixels that belonging together have similar color with no abrupt curvatures and normal changes are merged into the same region. New merged pixels are used as new seed until there are no pixels to comply with the condition. While RANSAC directly works on the 3D points clouds, it iteratively estimates

the parameters of a plane mathematical model and gives birth to plane parameters, which have the most inliers. In this study, these two planar region segmentations were tested and compared.

Part two of this study covers indoor scene 3D modeling using previous information. Figure 2 shows the process, which is also a continuation of Figure 1. By comparing the regional planes' normal and world coordinate axes, region alignment with the world axis is aligned. Each region is labeled with one of three colors, red, green or blue. Automatic floor detection is performed on a rotated points cloud, where the blue regions are parallel to the floor plane, which are parallel to the screen by rotation. Floor plane is detected as the plane with the smallest z values of (x, y, z) coordinates. Automatic wall detection is done with visibility of the potential walls at 2D projection plane of the rotated points. Then, the automatic wall boundary detection is performed based on the detected wall and floor boundary points. 3D modeling can be done with a domain image partition method.

The most important and basic concept of the study is the projective geometry and RGBD images. They are carefully illustrated in the following pages to prepare the reader for the following chapter's discussion.

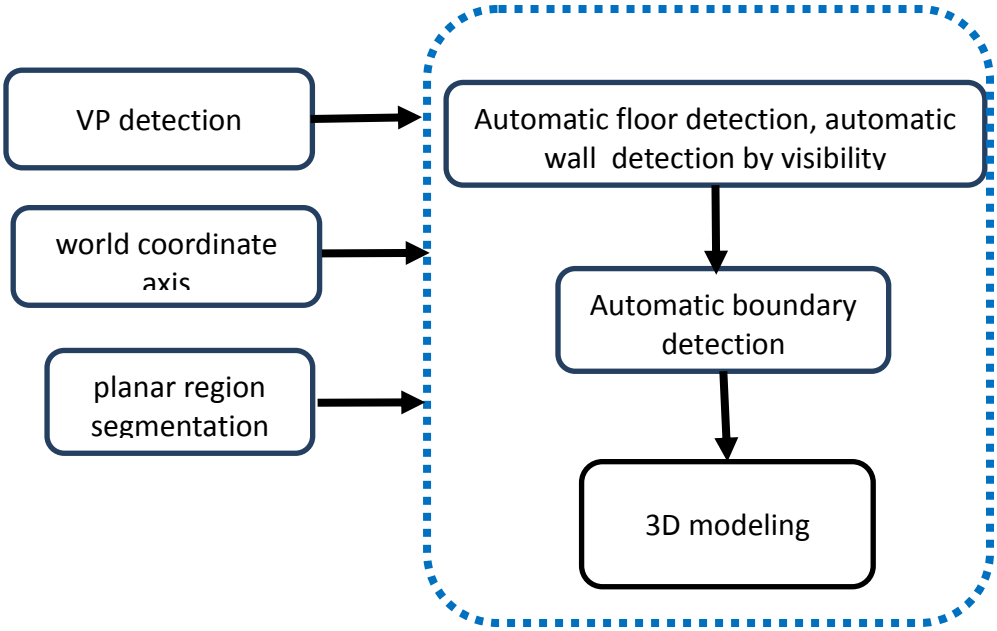


Figure 1. 2 is a continuation of Fig. 1.1, representing the final steps of 3D modeling.

1.3 **Perspective image formation.**

In order to do 3D modeling, understanding the perspective of each 2D image formed through the information captured through the pinhole and transformed into a demagnified 2D image is important. This discussion on perspective image formation used a pinhole camera in the modeling process [11, 12, 13]. The following equation describes the perspective projection image formation process.

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = P \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{1.1}$$

where λ is the arbitrary scaling factor. $(u, v,1)^T$ is the homogenous vector of pixel (u,v) in the image plane. $(X, Y, Z, 1)^T$ is the homogenous vector of the 3D world coordinate (X, Y, Z). P is the projection matrix, which is a 3×4 matrix with 11 unknown parameters. Figure 1.3 illustrates the perspective imaging.

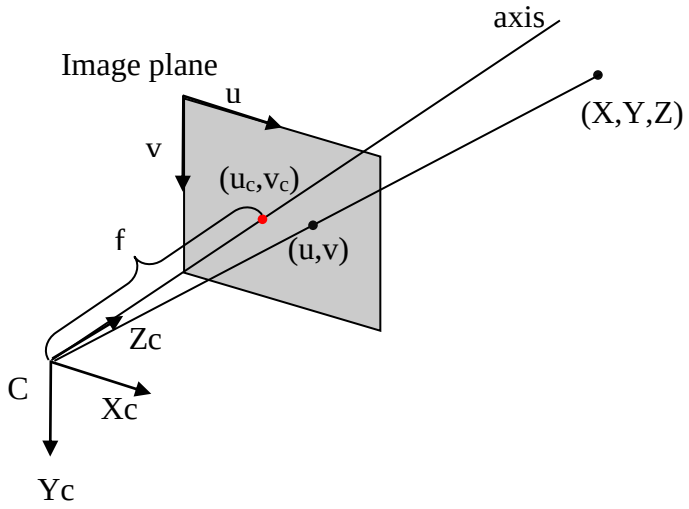


Figure 1.3 The perspective geometry. The Cartesian coordinate Xc ,Yc, Zc axis composes the camera coordinate. C is the camera center which is also the projection center. Cartesian coordinate Xw, Yw, Zw axis construct the words coordinate where world point (X,Y,Z) sits, which coincide with the camera coordinate. The u and v axis compose the image plane coordinate. In image plane, point (uc, vc) is the principle point which is the optical axis intersection point with the image plane. Every world point connects with C, and intersects the image at a point which is the perspective projection point, i.e., the imaging point. (X,Y,Z) is the world point under the camera coordinate system. (u,v) is the 2D coordinate under the image plane coordinate.

The mapping from the world coordinate, which coincides with the camera coordinate for now, to the image plane can be modeled as a simple P matrix as follows;

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{1.2}$$

where (X, Y, Z, 1) is the homogeneous coordinate of the 3D Euclidian coordinate of point (X,Y,Z). (x,y,1) is the homogeneous coordinate of the Euclidian coordinate of 2D plane point (x, y). Images projected on the image plane are usually sampled by 2D pixel arrays, and the image matrix origin is located at top left corner. Each pixel has a horizontal and vertical length kx and ky [length/pixel]. The digital image coordinate and the physical coordinate has the following transformation relation.

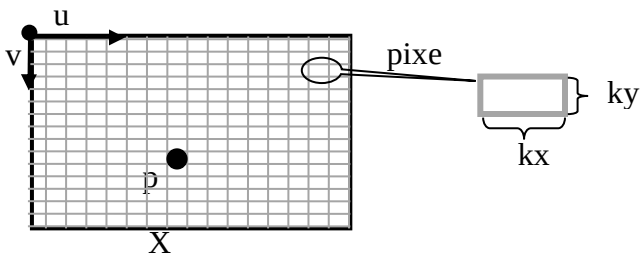


Figure 1. 4 The image coordinate system. P is principle point, which is not necessary at the image center. x and y have a physical length unit which is equal to Yc and Xc. point p = (uc , vc) is a principle coordinate at the image matrix coordinate system. Kx and ky are the physical length per pixel along the horizontal and vertical directions. Thus $x/kx=u-u_C$;$y/ky=v-v_C$

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & u_c \\ 0 & f_y & v_c \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = K \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{1.3}$$

where $f_x = f/kx$ and $f_y = f/ky$ represent the focus length in terms of pixels. Usually, we have kx = ky; The above K 3 x 3 matrix is called the camera calibration matrix, which is also referred to as

the intrinsic matrix. This intrinsic matrix is composed four parameters: scaling in the u and v directions α_x, α_y. The principle point is (u_c , v_c), which is the point where the optical axis intersects the image plane. When the world coordinate system does not coincided with the camera coordinate system, there exists a rotation and translation transform from world coordinate to camera coordinate.

$$\begin{pmatrix} X' \\ Y' \\ Z' \\ 1 \end{pmatrix} = \begin{pmatrix} R & t \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{1.4}$$

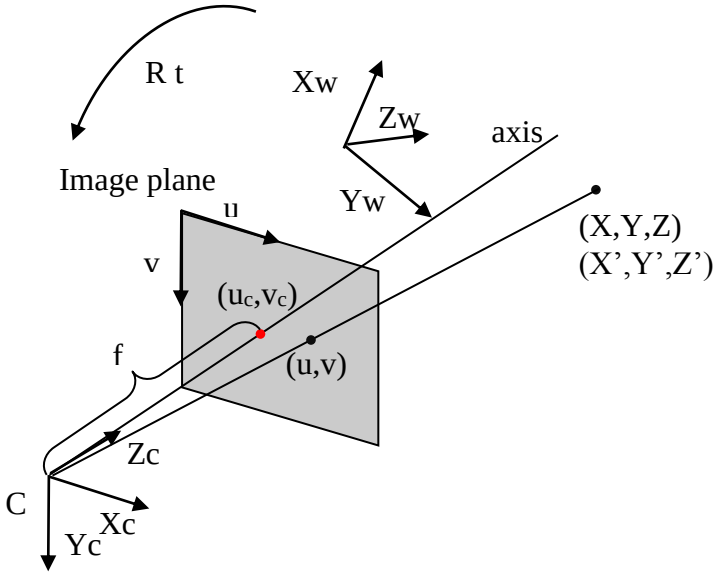


Figure 1. 5 Defining the transform from world coordinate to camera coordinate by rotation R and translation t.

In Figure 1.5, R is a 3 x 3 rotation matrix, and t is a 3 x 1 translation vector. Concatenating the three matrices, we obtain the following projection model.

$$x = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} R & t \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = K\left(R \mid t\right) X = PX = \begin{pmatrix} \pi_1 & \pi_2 & \pi_3 & \pi_4 \end{pmatrix} X \tag{1.5}$$

which define the 3 x 4 projection matrix P. The six external parameters, which form the rotation R and translation t, are three rotation and three translation parameters. The pinhole model is an ideal imaging model where there is no optical aberration, In reality, many aberrations exist in the image due to the imperfect lens design and subsequent misalignment. Usually only the third order optical aberrations are considered, which are classified into spherical, coma, astigmatism, curvature, and distortion. In photometry, only distortion affects the metric measurement; thus, the distortion is usually modeled. It is done before K and after projection.

From Equation (1.6) we have the ideal projection point m. md is the distorted point. On the right, kc is the radial distortion coefficient, Δ is the tangential distortion caused by optical misalignment. (xd,yd) is the distorted image points as projected on the image plane, while (x, y) is the undistorted image point as projected by an ideal pinhole camera. (u0,v0) is the principal point. Kn is the radial distortion coefficient, and Pn is the tangential distortion coefficient [14, 15].

The preceding distortions can be seen in

$$
\begin{aligned}
m &= \begin{pmatrix} Xf/Z \\ Yf/Z \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \\
m_d &= \begin{pmatrix} x_d \\ y_d \end{pmatrix} = m\left(1 + k_{c1}r^2 + k_{c2}r^4 + k_{c5}r^6 + \cdots\right) + \Delta \\
\Delta &= \begin{pmatrix} \left(2p_1xy + p_2(r^2 + 2x^2)\right)\left(1 + p_3r^2 + p_4r^4 + \cdots\right) \\ \left(2p_2xy + p_1(r^2 + 2y^2)\right)\left(1 + p_3r^2 + p_4r^4 + \cdots\right) \end{pmatrix}
\end{aligned}
\tag{1.6}
$$

Hence, when the extrinsic and intrinsic parameters are determined and all the distortion parameters are obtained, the camera calibration is done.

1.4 **Vanishing points**

In man-made environments such as in suburban scenes, architectural structures, and indoor scenes, parallel and orthogonal edges exist everywhere [16]. When imaged by perspective projection, the straight edges are still straight edges. However, perspective projection doesn't reserve angles. Parallel straight edges are not parallel anymore, but converge to a point called the vanishing point on the image plane. Orthogonal edges' angle is also not 90 degrees any more. Vanishing points lying on the same plane in the scene form a line in the 2D image, which is called vanishing line. Vanishing points and vanishing lines can be used to refer to the calibration and provide important cues for 3D modeling. Figure 1.6 illustrates the vanishing points and vanishing line of a cubic object. The modeling and understanding of the indoor scene can be simplified with the help of vanishing points. For instance, the green wall in Figure 1.6 has two red edges converging to vanishing point C. Thus, leveraging the vanishing point C and any point on the red edges, we can find the green plane's extension horizontal limit in the projective image

shown as red dash lines. Similarly, other walls' horizontal range can be found by the same method.

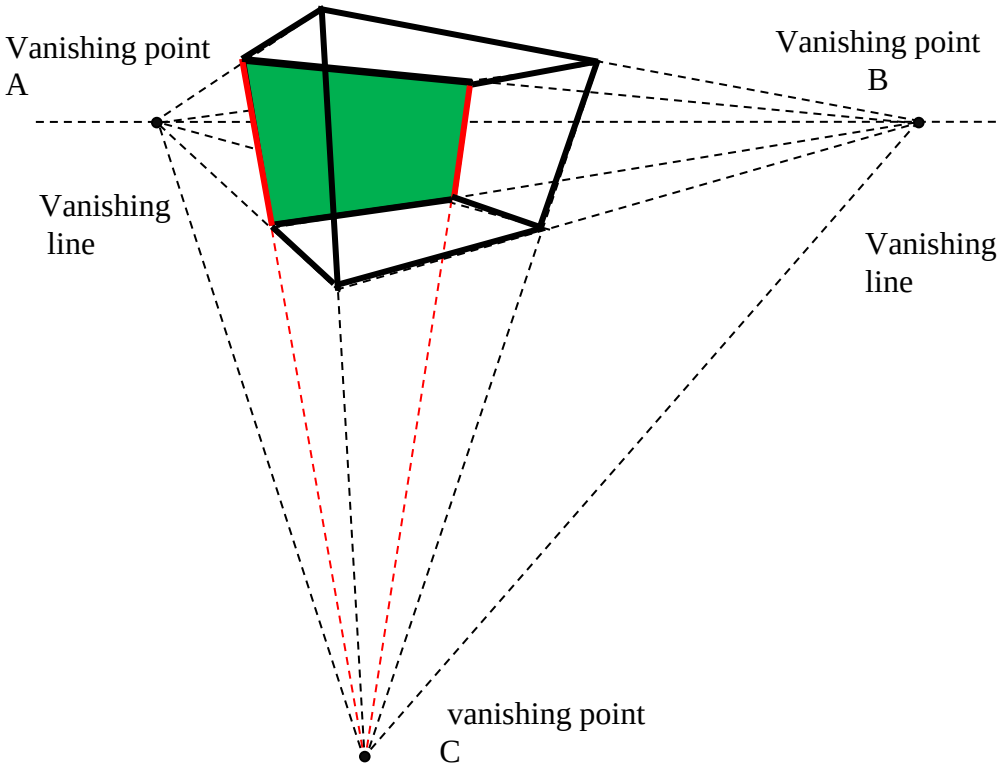


Figure 1. 6 Vanishing points and vanishing lines in 2D perspective image. Parallel lines' perspective projections converge at vanishing points. Connecting vanishing points forms vanishing lines.

For another instance, indoor scene world axes defined by the wall, and ground edges lines converge to three vanishing points. Figure 1.7 illustrates the geometry. The world axis X_w,Y_w,Z_w are aligned with edges of ground and walls. So these axes converge to corresponding vanishing points as other edges. X_w,Y_w,Z_w, the red, green and blue axis converge to VP1, VP2, VP3 points, respectively. Vectors $\overrightarrow{CVP1}$, $\overrightarrow{CVP2}$, and $\overrightarrow{CVP3}$ are parallel to the axes X_w,Y_w,Z_w. Thus, $\overrightarrow{CVP1}$,

$\overrightarrow{CVP2}$, and $\overrightarrow{CVP3}$ are orthogonal to each other. The normalized vector gives the world axis estimation [16,17].

$$\left(X_w, Y_w, Z_w\right) = \left(\frac{\overrightarrow{CVP1}}{\left\|\overrightarrow{CVP1}\right\|} \quad \frac{\overrightarrow{CVP2}}{\left\|\overrightarrow{CVP2}\right\|} \quad \frac{\overrightarrow{CVP3}}{\left\|\overrightarrow{CVP3}\right\|} \right) \tag{1.7}$$

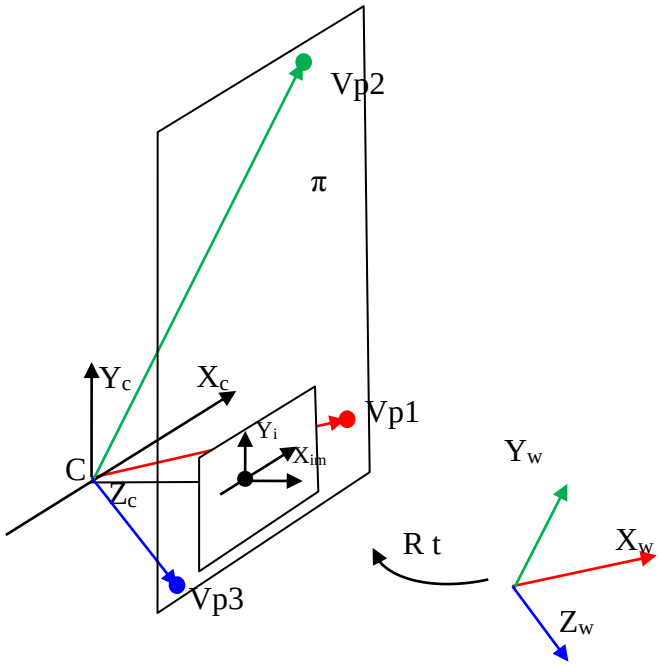


Figure 1. 7 World coordinate axes defined parallel to the indoor scene edges, like the X,Y,Z world axis. Hence, these axes converge to corresponding vanishing points. X, Y, Z, the red, green and blue axis, converges to VP1, VP2, VP3 points. Lines CVP1, CVP2, and CVP3 in the camera frame are parallel to X, Y, Z axes. Plane π includies vanishing points is an augmented image plane.

Common routines for detecting vanishing points usually include two steps: 1) To enable line detection in the image, one of the mostly commonly used methods is first canny edges detection and then line segment linking. 2) Calculate the intersection points of these lines as

vanishing point candidates. Use some optimal methods like least square, or voting method to derive the optimal vanishing points. In chapter 2, a novel method used in this study is presented.

1.5 **RGBD image.**

RGBD imaging is becoming an important imaging type thanks to current development of several RGBD sensors. Kinect is among the most important indoor scene 3D sensing devices due to its low cost, reliability, and speed of measurement. Here, we give a geometrical analysis, a geometrical model, and calibration before discussing its performance.

In Kinect [19, 20, 21], an IR speckle projector projects a fixed pseudo random pattern, an IR camera, which captures the scattered speckle pattern of the object, which uses the distorted speckle pattern to triangulate the points in the world coordinate. IR sensor has 640 × 480 pixels at 30 fps, or 1024 × 768 at low frame rate. Field of view (FOV) is 57×43 degrees. The focal length is 6.1 mm, and the pixel size is 5.2μm. It captures IR patterns scattered by the object. The captured image correlates with a reference image, which is obtained by capturing a plane at a known distance from the IR camera, Decoding the pattern and the result of the correlation creates a disparity value (depth) for each pixel. If the projector is blocked, the IR camera can be calibrated with a checker board calibration target illuminated by a halogen lamp using the same camera calibration routine. The speckle projector and IR camera compose the depth sensor. A color camera is used to image color content and provide color texture 3D points. The color camera has 640 × 480 pixels at 30 fps and 1024 × 768 at low frame rate. It also has an FOV of 63 × 50 degrees, a 2.9 mm focal length, and a 2.8 μm pixel size. Figure 1.8 illustrates the Kinect components. As an imaging device, Kinect provide color image and inverse depth image (1, 2, 20). Rather than directly providing the depth z, Kinect gives inverse depth image. We adopt a correction equation to obtain depth z.

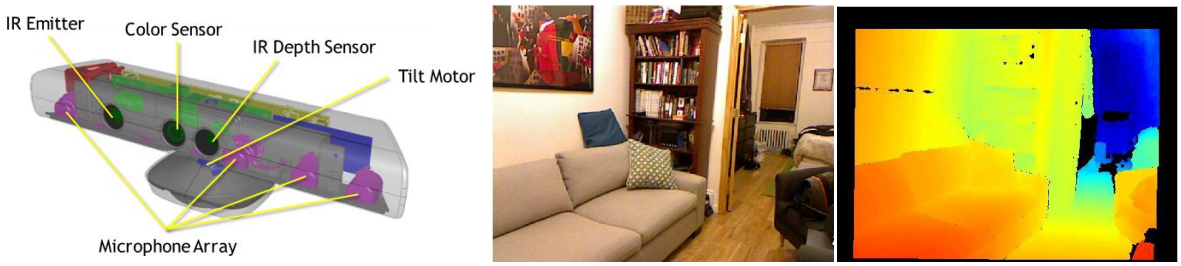


Figure 1. 8 Kinect, (courtesy of Microsoft, Inc.), color image and depth image.

The depth z range is from 0.5 to 15 m. The depth resolution changes with depth, which can be qualified with the following function of the depth z. $q(z) = 2.73z^2 + 0.74z - 0.58[mm]$ where z is from 0.5 to 15 m. We can see from the equation, q (0.5m) = 0.65 mm. q (15 m) = 685 mm.

Kinect can be modeled as a multi-view stereo vision system; the geometrical model is shown in Figure 1.9. IR camera and projector triangulate 3D points (X,Y, Z) to get depth image. The same point is also captured by the color camera. Depth image and color image have a rotation and translation transformation (R T) shown in the figure. The models of cameras are similar to that of Section 1.3, but with a lower order of distortion coefficient.

$$\begin{pmatrix} u & v & 1 \end{pmatrix}^T = K_{C/IR} \begin{pmatrix} x_d & y_d & 1 \end{pmatrix}^T$$

$$\begin{pmatrix} x_d \\ y_d \\ 1 \end{pmatrix} = \left(1 + k_{c1}r^2 + k_{c2}r^4 + k_{c5}r^6\right) \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} + \begin{pmatrix} \left(2p_1xy + p_2(r^2 + 2x^2)\right)\left(1 + p_3r^2 + p_4r^4\right) \\ \left(2p_2xy + p_1(r^2 + 2y^2)\right)\left(1 + p_3r^2 + p_4r^4\right) \\ 1 \end{pmatrix} \quad (1.8)$$

where $K_{C/IR}$ is a color camera or an IR camera intrinsic matrix, k_{c1}, k_{c2}, k_{c5}, and p1,p2 p3 p4 are radial and have tangential distortion parameters of a color camera or IR camera. $(x,y,1)^T$ is the projected ideal point. The depth camera produces an inverse depth measured from the front of the IR sensor, d'= f(1/z). The depth image could be corrected by the following equation, d =

f(d'). With a depth image (u,v,d) and calibration matrix, the 3D point (X,Y,Z) can be obtained by

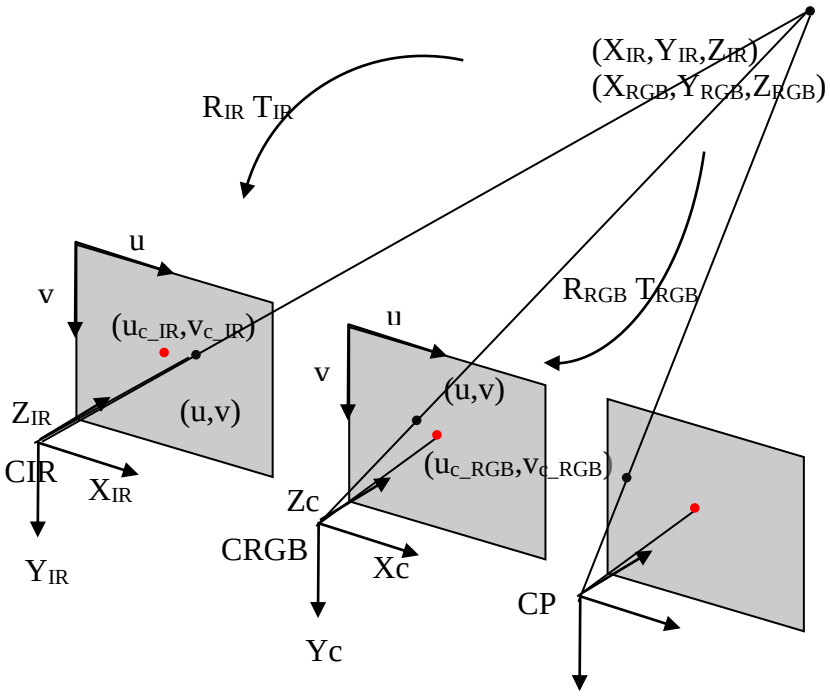


Figure 1. 9 Geometrical model of Kinect. CIR is the center of the IR camera. CRGB is the center of the color camera, and CP is the center of projector.

The 3D point can be transformed to an RGB frame and projected to an RGB camera with

$$\begin{pmatrix} u_C \\ v_C \\ 1 \end{pmatrix} = K_C disto(RM + T, k_c, p_c). \tag{1.9}$$

This equation gives depth to color image mapping, which could align the two images. In our real processing, we ignore the distortion of the color camera for simplicity. The depth image and color image are mapped together by the above equation.

CHAPTER 2. RGB IMAGE PROCESSING

2.1 Depth to RGB mapping and hole filling

According to Section 1.4, we know that the Kinect gives an inverse depth image referring to an IR camera frame. We adopted the following correction equation. The real depth is corrected with this equation.

$$d_{abs} = 0.1236 \tan\left(\frac{d_{inv}}{2842.5} + 1.1863\right) \tag{2.1}$$

where d_{inv} is the given inverse depth pixel value which is a 11 bits values with a range of 0–2047. dabs is the absolute depth value in meter units. A threshold is applied to limit the measured depth range (depthmax depthmin) = (0 10 m). A corrected depth image is shown in Fig 2.1.

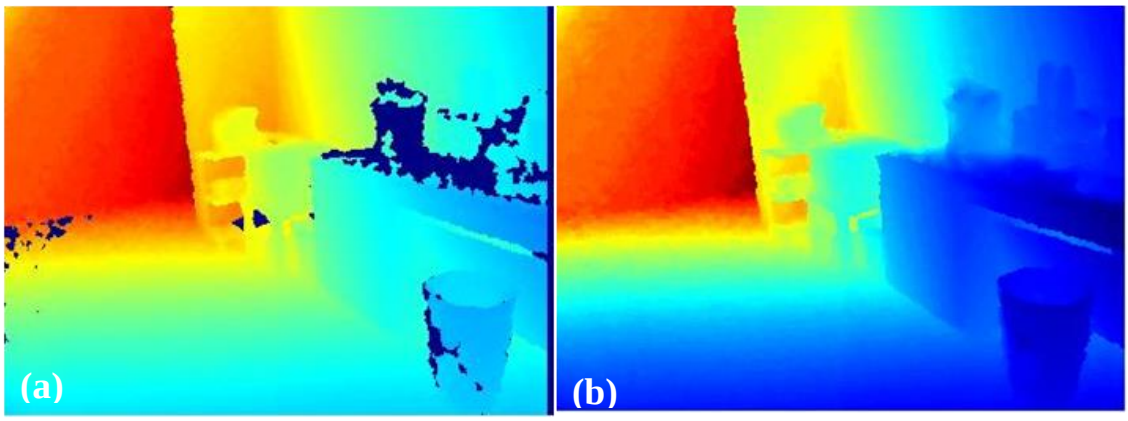


Figure 2. 1. Depth Images: a) original relative depth image and (b) corrected absolute depth image.

We could see a lot of holes in the depth image. The holes may be caused by a missing reflectance signal of special surfaces, or it may be caused by wrong-decoded speckle patterns. No matter what kind of reasons causing the holes, we need to fill them because they will cause modeling mistakes. Color image provides hints on how the filling goes. In order to aid the

filling, we register the depth image to an RGB image by first transforming the depth image to a 3D points cloud in the frame of an IR camera. This was done by the following.

$$\begin{pmatrix} X_{IR} \\ Y_{IR} \\ Z_{IR} \end{pmatrix} = \begin{pmatrix} \left(u - u_{c\_IR}\right) d_{abs} / f_{x\_IR} \\ v - v_{c\_IR} / f_{y\_IR} \\ d_{abs} \end{pmatrix} \tag{2.2}$$

Then the point is transformed to the RGB camera frame by the following

$$\begin{pmatrix} X_{RGB} \\ Y_{RGB} \\ Z_{RGB} \end{pmatrix} = R_{RGB} \begin{pmatrix} X_{IR} \\ Y_{IR} \\ Z_{IR} \end{pmatrix} + T_{RGB} \tag{2.3}$$

Then, these 3D points are projected onto the color image plane by the following.

$$\begin{aligned} \lambda \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} &= K_{RGB} \begin{pmatrix} X_{RGB} \\ Y_{RGB} \\ Z_{RGB} \end{pmatrix} \\ \lambda \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} &= Z_{RGB} \begin{pmatrix} X_{RGB} * f_{x\_RGB} / Z_{RGB} + u_{c\_RGB} \\ Y_{RGB} * f_{x\_RGB} / Z_{RGB} + v_{c\_RGB} \\ 1 \end{pmatrix} \end{aligned} \tag{2.4}$$

where λ =ZRGB is the depth factor. The projected pixel is not an integer, so prime is added for discrimination.

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \min \left(round \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} \right) \tag{2.5}$$

The 3D point M is first projected onto the color image plane by (u',v',1)= Kc (RcM+Tc), projection location (u',v') may not be an integer within the color image range [1…M;1…N], where M and N is the columns and rows of the color digital image. so round (u',v') will round into a real pixel location. Considering this from another side, there may be multiple projections

at single pixel location, so the points with the shortest z is reserved. This is similar to z-buffer processing in graphics. We use min to represent z-buffering. For pixel (u',v') , a unique depth value is mapped onto it. Due to the angle and the sensor size difference between the IR camera and color camera, the mapped depth image may not cover every pixel of color image. The non-overlapping part must be cropped. We reserved [m…M-w; n…N-h] section of color image and mapped depth image. Attention was also be directed to the principle point of color image planes, which should be shifted by (m,n) , i.e. (uc',vc') = (uc_RGB-m,vc_RGB-n) since we have cropped the images. So the updated Kc is given as follows

$$K_C = \begin{pmatrix} f_{x\_RGB} & 0 & u_{c\_RGB} - m \\ 0 & f_{y\_RGB} & v_{c\_RGB} - n \\ 0 & 0 & 1 \end{pmatrix} \tag{2.6}$$

Fig. 2.2 Mapped depth image in color image.
Left: Before mapping. Right: After mapping

Figure 2.2 shows a mapped depth image onto a color image plane. We can see similar holes existing in the depth image. However, the depth image overlaps with the mapped color image providing hints to fill holes. A bilateral filtering method with color image was adopted to fill holes. Bilateral filters can estimate the depth information of holes but reserves edge information; thus, it tends to give correct depth information. Figure 2.3 presents a before and after picture with the second image representing the effect of hole filling.

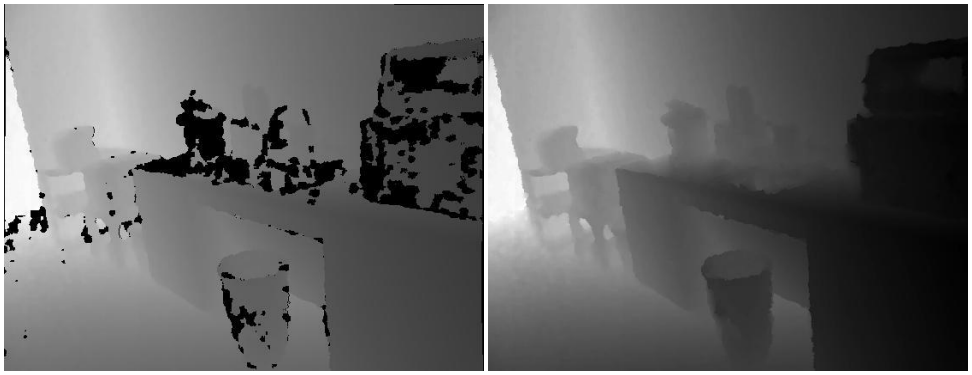


Figure 2.3 Depth image before hole filling and after hole filling.

2.2 3D point cloud.

With the mapped depth image filled, the complete 3D points are ready to be reconstructed. The following equation is used to transform depth to 3D points in an RGB frame.

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} d_{mapped}\left(u - u_{c\_RGB} + m\right) / f_{x\_RGB} \\ d_{mapped}\left(v - v_{c\_RGB} + n\right) / f_{y\_RGB} \\ d_{mapped} \end{pmatrix} \tag{2.7}$$

With color image we have color textured 3D point clouds (RGB(u,v)+XYZ). The following figure shows four RGBXYZ point clouds.

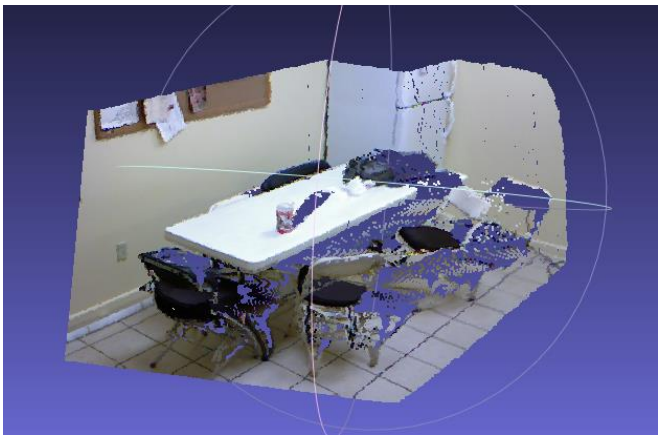


Figure 2. 4 3D point clouds with color texture.

2.3 Detection of vanishing points and their application in layout estimation and world coordinate estimation

As shown in Section 1.4, a human-made scene usually has parallel lines and edges, which appear as straight lines in 2D image and converge to vanishing points. Correspondingly, vanishing points have an application in 3D modeling. One of the applications is to use vanishing points to calculate the unite vector of the world coordinate (nx,ny,nz). World axis can be used to label planes that are aligned with these axes, thereby enabling categorization of planes and definition of the relations among planes. The second application is to find adjacent walls' vertical margin. Its feasibility is due to the fact that vertical margins are shown as converging lines, which converge to a vanishing point in a 2D image. In turn, the lines connecting the vanishing point and one of the edges' points correspond to margin lines of walls. These usages will be explained in chapter 4.

The first step to finding vanishing points in a 2D color image is to detect lines. The line detection method used here is adapted from [22]. Local straight line segments are image regions where the local gradient and level lines of the image are greatest. The method first computes the level-line angle at each pixel to produce a level line angle field, i.e., a united vector field such that all vectors are perpendicular to the gradient orientation through their base point. Next, the field is segmented into pixel regions, which share the same level line angles up to a certain tolerance threshold τ. The pixel regions are called line support regions and are subject to a validation process. Each line support region is a candidate of a line segment. The regions that survive the validation process are retained as a line segment. The corresponding geometrical structure associated with it is a rectangle. The principle inertial axis of the line support region is the main line segment direction. The pixels in the support region with level line angles within the tolerance τ are called aligned points. The total number of pixels n within the rectangle and its

number k of aligned points are compared to provide validation for the line segment as noted by [22].

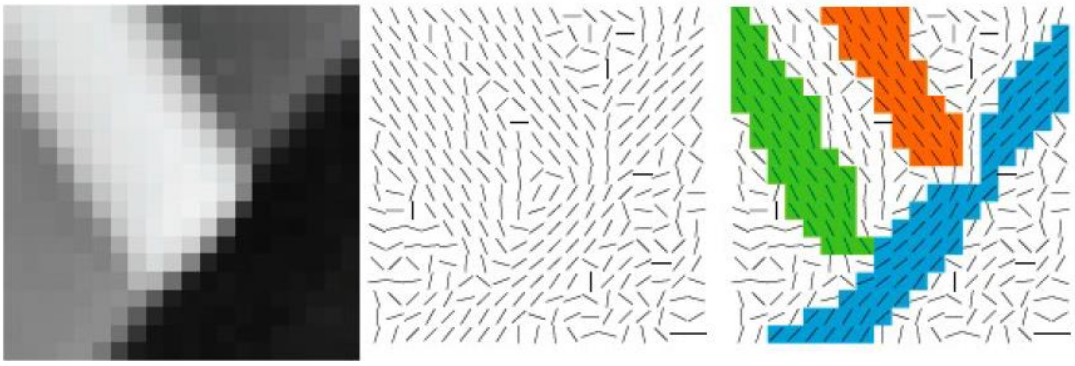


Figure 2. 5 Left: input image; middle: level line filled; right: line support regions

After line segments are detected, the vanishing points can be derived either by finding the centroid of the intersection points of line segments, or by finding the highest vote in the Hugh transform domain [23]. A similar method is given by Dubska et al. [24]. A dual image space (called a parallel coordinate (PC) space) is created. PC spaces were proposed by Maurice d'Ocagne and popularized by Alfred Inselberg [25] and are usually used as a multi-variable data visualization tool. Dubska et al. used the parallel coordinate to detect lines, which have the similar concept to that of Hough transform. The points in the original image are represented as lines in the PC domain. Considering the dual properties of lines and points of perspective geometry, lines can be represented as points in a PC space domain. We thoroughly utilized both the image domain and PC domain to explore transformation properties of points and lines in the context of vanishing point detection.

Usually, N dimensional data in Euclidean space is represented by Cartesian coordinates. However, it is difficult to visualize > 2 dimensional data on two dimensional objects. Parallel coordinates represent the data by parallel axis with each axis representing one of the dimensions, which can visualize any dimensional data. N-dimension data is represented by n-1 lines

connecting n axis points on the parallel axis. In 2D images, a point can be represented by homogeneous coordinates. (x, y, w) where (x, y) is the corresponding Cartesian coordinates of Euclidean space. The point is represented as lines in PC. Collinear points intersect at a single point. Figure 2.6 shows an example of points to lines mapping between two dimensional coordinates system.

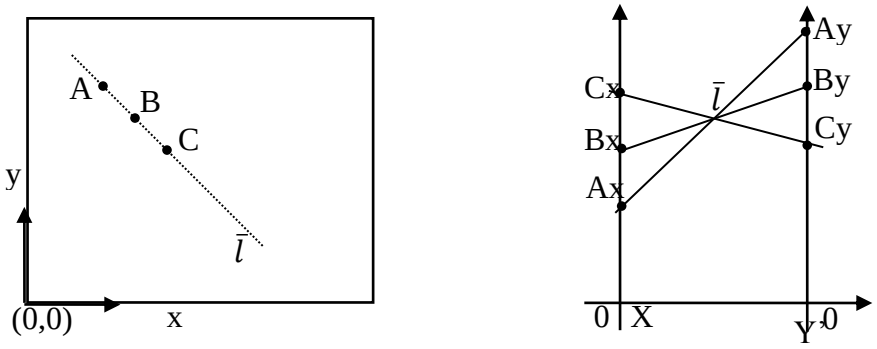


Figure 2. 6 Left: the original perspective geometry plane, x and y are coordinates.

X coordinates points A, B and C and are shown as vertical axes x' of parallel coordinates on the right side. Y coordinates of the points are sitting at vertical axis y' of parallel coordinates. X and x' have same values, and y and y' have the same values. Right: parallel coordinates x' correspond to x coordinates of the original image. y' corresponds to y coordinates of the original image. A, B and C are mapped to lines $\overline{AxAy}$, $\overline{BxBy}$ and $\overline{CxCy}$. Due to duality of lines and points, point in perspective coordinates are mapped to lines of parallel coordinates. A line in a perspective coordinate is mapped to the points of a parallel coordinate.

As shown, points in the x-y coordinate are represented as lines in the parallel coordinate in two dimensional cases. The intersect points $\bar{l}$ of the mapped lines in the parallel coordinate correspond to a line that passes through the mapping collinear points of the perspective

coordinate. Some special lines, like y=x , are mapped to infinite points, which are the vanishing points in parallel space. Perspective geometry representation of the lines can handle this case. For a line ax+by+c=0 whose homogeneous coordinate is [a, b, c], the perspective geometry begins by being mapped to points $\bar{l}$ by the following equation

$$l:[a,b,c] \Rightarrow \bar{l}:[db,-c,a+b] \tag{2.8}$$

where d is the distance between x' and y'; This equation give the 2D points in the parallel coordinate as [db/(a+b),-c/(a+b)]. $\bar{l}$ is in-between x' and y' only if $\infty > a/b > 0$ for a+b = 0. The points are at infinity. For a=0, $\bar{l}$ sits on the y' axis. For vertical lines, i.e., $a/b = \pm\infty$, $\bar{l}$ sits on the x' axis. In order to represent lines $-\infty < a/b < 0$, a twisted space T, composed of coordinate axes x' and -y' is presented in this thesis. The x'and y' region is called a straight space S. the T space generation has the same x' coordinates to the S space except the –y' is an invert of y' lines, that is, $-\infty < a/b < 0$ is mapped in-between x',-y'. Considering that T and S space have the same x', T and S space can be combined as shown in Figure 2.7.

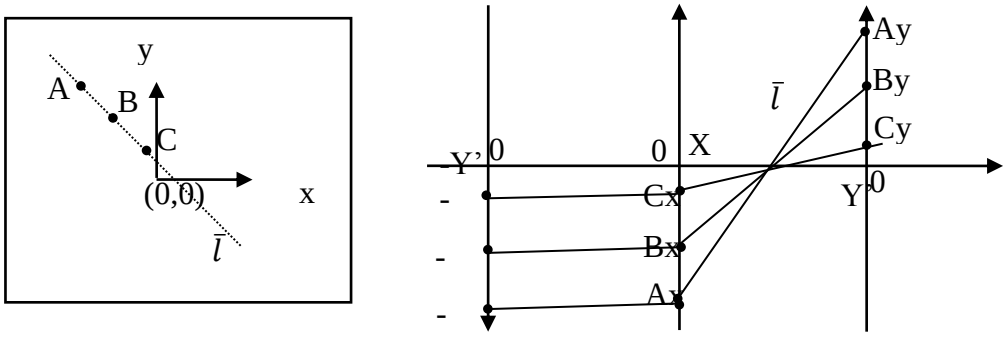


Figure 2. 7 Right: x and y center shifted Euclidian coordinates. Left: Twisted and straight spaces. Center shifted coordinates benefit the calculation.

The mapping between two spaces becomes:

$$
\begin{aligned}
l:[a,b,c] &\Rightarrow \bar{l}_S:[db,c,a+b] \quad \text{if } 0 \le b/a < \infty \\
l:[a,b,c] &\Rightarrow \bar{l}_T:[-db,c,b-a] \quad \text{if } -\infty < b/a \le 0
\end{aligned}
\tag{2.8}
$$

Eq. 2.8 is based on the property that a line in the TS space corresponding to a point in the x-y space can generate mapped points $\bar{l}$ lie on single line if lines l intersect in one point in the x-y space. Points in the x-y space collinear if mapped lines in x'-y' intersect at one point. Vanishing points which are at the intersection point of perspective lines are also parallel in a real 3D world. These lines are also mapped to a line in TS space. The line intercepting points on axis x', y' or –y' give us the vanishing points coordinate.

Similarly, the collinear points' line mapping in TS space intersect at one point which correspondingly gives us the line equation in x-y space. Instead of finding vanishing points, the line segments detected in step 2 is the input. End points of segments are mapped to line segments in TS space. Local maxima aggregation points are where multiple points lie in a collinear position (a straight line) in the x-y space. The local maxima points are using the sliding window technique and finding the pixel with maxima votes is like finding local maxima in the Hough transform space. Every local maxima is mapped back to a line of x-y space where multiple points collinear. The input line segment endpoints often represent some collinear structure or repetitive structure of a human-made environment. Thus, a local maxima search can help to find these line structures.

The line segments themselves can be mapped to points in TS space. Lines segments that intersect at a vanishing point are mapped to collinear points in TS space. To find vanishing points, line segments were mapped to points in TS space first. A multiple line fitting method based on RANSAC was developed to fitting lines on these points. The fitted lines intersecting the x' and y' or –y' generated the vanishing point candidates. The points belonging to a line indicate which

lines in the x-y space intersect to common vanishing points. In this way, the line segments can be classified according to the vanishing points they are associated with.

2N line segment endpoints were obtained from the N line segment detection step. Before starting, two coordinates are defined: the perspective coordinate and the homogeneous coordinate. The perspective coordinate in the image is defined as the origin in the center of an image. The horizontal x axis points to the right, the vertical y axis is up-right, the pixel order (i, j) defines the Euclidian coordinate (x, y) = (i,j). Augmenting the coordinate to (x,y,w) with w=1 produces the homogeneous coordinate. The perspective coordinate in parallel coordinate space is also defined, where u and v define the domain coordinate. -y' axis and x' axis form the twisted space, x' and y' form the straight space. The distance between them is d. x' coincides with u, while x', y', and -y' are all perpendicular to v. The homogenous coordinate is formed by augmenting the coordinate (u, v) to (u ,v ,w') with w=1 here. Each segment produces two endpoints. Endpoints are represented with homogeneous coordinates (x_i, y_i, w), x_i / w and y_i / w are the points' Euclidian coordinates. Next, the points are mapped to TS space by locating point $(0, x_i / w, w')$ on x' axis, point $(d, y_i / w, w')$ on y' axis, $(-d, y_i / w, w')$ on -y' axis, where x' and y' comprise the straight space, i.e., S space of PC space, x' and –y' comprise the twisted space of PC space, d is the distance between adjacent parallel axis. Connecting points $(0, x_i / w, w')$ and $(d, y_i / w, w')$ forms line l_{Ti} in T space; points $(-d, y_i / w, w')$ and $(0, x_i / w, w')$ form line l_{Si} in S space. Thus each endpoint is mapped to three points on parallel axes, two line segments in T and S space. All the points are mapped to the TS space, which give us the complete transform.

The following figure shows an example of VP detection. (a) is the detected line segments; (b, c) shows the straight and twisted PC space voting from the line segment of (a). Lines fitted in the

PC are transformed back to vanishing points. (d) Only the line voting vanishing points are reserved.

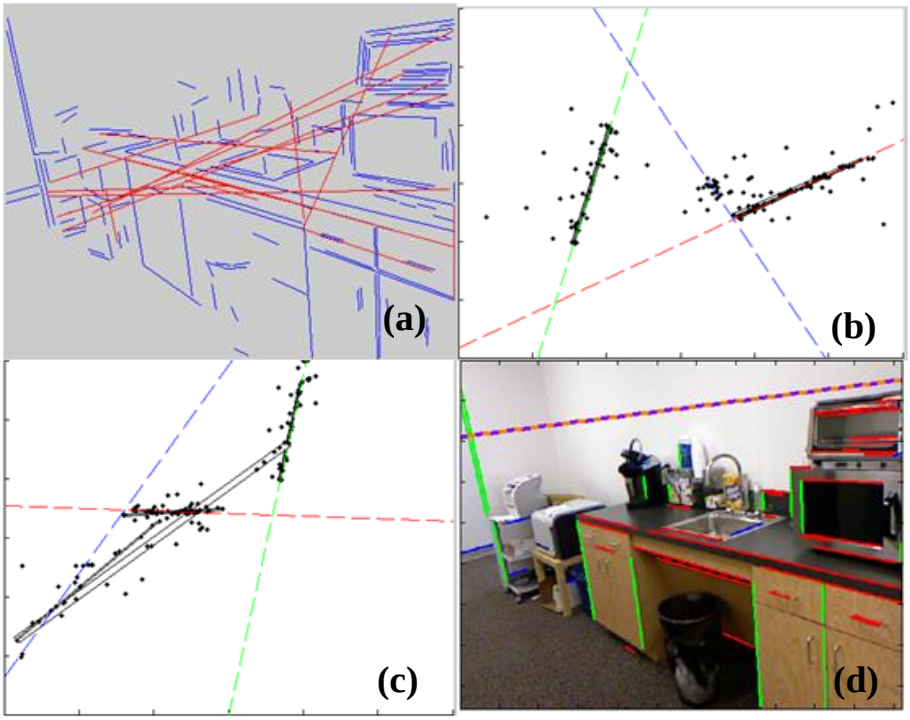


Figure 2.8 The vp detection. a. the line segment detection. Only the linked lines are reserved. Figures 2.8b and c shows the voting points of these lines in PC domain. Hence, Figures 2.8b and c show straight and twisted space respectively. Lines are fitted and transformed back to points of image space. The points are vanishing points. Linking vanishing points gives a horizon line in Figure 2.8d.

Table 2. 1 The vanishing point and world axis

	R	G	B
Vpx	-179.732288765201	496.263843971930	1031.08171098709
Vpy	117.342192401043	2112.85714598943	-34.2724019459170
Nx	-0.685643157735695	0.185068013889734	0.704019382180018
Ny	0.088084586858112	-0.959447887296938	0.267770157260808
Nz	0.765270158498055	0.312409062206595	0.562816277628462

Acoording to Figure 1.7 and Equation 1.7, vanishing points (VPs) in the camera frame form vectors that are parallel to the world axis. VPs are converted to unite vectors of world axis

by using the camera's calibration parameters. Equation 2.8 gives the transformation, where u_{VP} and v_{VP} are the vanishing point coordinates in the image plane. Since the Kinect RGB sensor basically has equal $f_{x\_RGB} \approx f_{y\_RGB}$ in terms of pixel units, we average them to get z_{VP} coordinate in the sensor frame. Normalizing x_{VP}, y_{VP}, and z_{VP} gives the world axis vectors shown as red , green and blue shown in Table 1.

$$\begin{pmatrix} x_{VP} \\ y_{VP} \\ z_{VP} \end{pmatrix} = \begin{pmatrix} u_{VP} - u_{c\_RGB} + m \\ v_{VP} - v_{c\_RGB} + n \\ \left(f_{x\_RGB} + f_{y\_RGB} \right) / 2 \end{pmatrix} \tag{2.8}$$

$$\begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} = \begin{pmatrix} x_{VP} \\ y_{VP} \\ z_{VP} \end{pmatrix} \Bigg/ \left\| \begin{pmatrix} x_{VP} \\ y_{VP} \\ z_{VP} \end{pmatrix} \right\| \tag{2.9}$$

CHAPTER 3. PLANE DETECTION

3.1 Introduction

With only an RGB image, it is difficult to reconstruct 3D. 3D cues are primarily found from vanishing points. However, with RGBD data, the difficulty of modeling can be eased significantly. Plane detection in point clouds is a prerequisite to 3D modeling. In computer vision, RANdom SAmple Consensus (RANSAC) [10] is one of the most widely used methodologies for plane detection. It is immune to the presence of outliers, and it has proven to be successful in both 2D and 3D data sets. Region growing has been adopted in point clouds. Region growing has a large amount of implementations in processing 2D image and 3D volumetric data. In this study, we reinvent a region growing method with joint RGB and 3D point clouds. The algorithm is presented in Section 3.2. In Section 3.3 we show the growing results, and Section 3.4 shows RANSAC plane fitting results for comparison.

3.2 Planar region growing with RGBXYZ data.

Region growing [26, 27, and 28] is a mature, simple region-based image segmentation method. It uses initial seeds to start the pixel based regional growing. Iteratively, it examines neighboring pixels of the seeds and decides if the neighboring pixels should be added into the region or not depending on a regional membership criterion.

The goal of region growing is to partition the Image I into multiple regions {Ri: i=1…N} so that 1) $I \leq \bigcup_{i=1}^{N} R_i$ with '=' represents complete segmentation, 2) Ri is an independent, connected region, i.e., $R_i \cap R_j = \emptyset$, with $\emptyset$ being the null set if $i \neq j$. "$\leq$" means incomplete segmentation, since initial seeds may not be sufficient for complete segmentation. By selecting

enough seeds, complete segmentation can be achieved. Every pixel could be assigned to a region but only to one region.

The first step is to select a set of initial seeds points, as the name suggests. Seed selection depends on some costumed criteria, such as pixels defined on a grid, or pixels having some special features or properties. For our modeling, region growing is to partition the image into planar regions; hence, the seeds should be selected within planar regions. These kinds of seeds can be obtained from some initial, rough segmentation or from points with minimum curvature in the point cloud or user inputs.

The region grows from one of the seeds to neighboring points based on the regional membership criteria. RGB image and point clouds both provide a large amount of cues for planar region growing. These cues can be used for creating the criteria. For example, a plane tends to have similar color, and a pixel with a very different color should not be added to the region. Thus, color difference can be a criterion. For comparing colors, the RGB image was firsy transformed to CIE-Lab color space image. Among the growing iteration, threshold T_c and color difference determine whether any neighboring pixel could be added to regions and become a new seed or not.

$$\sqrt{(a_i - a_n)^2 + (b_i - b_n)^2} < T_c$$

The color difference (delta E) is often use for comparing colors, which is measured as the difference between L, a, and b values. We ignore L, brightness, because brightness changes with the viewing angle. The difference values of a and b components are more stable.

The point clouds provide another two criteria. The first is normal criterion where points in a plane tend to have close normal vectors. Angles between these normal vectors are close to zero. Point normal is first calculated with neighboring points as shown in Figure 3.1. The

neighboring points can be found by either K Nearest Neighbor (KNN) with fixed distance, or KNN with a fixed number of neighbor points [29]. For achieving fast speed indexing, a kd-tree was built upon point cloud 31. Details are in Appendix 2. K nearest neighbor researching with KD trees. Figure 3.1 shows query point's P_{query} enclosed with k nearest neighbors, which can be expressed as

$$p = \left\{ p_i \mid \left\| p_i - p_{query} \right\| \le R_{query} \right\}$$

p_i is the neighboring points, ||.|| is Euclidian distance. The plane that fit these points gives the normal estimation.

$$M = \frac{1}{k}\sum_{i=1}^{k}\left(p_i - \bar{p}\right)\left(p_i - \bar{p}\right)^T$$
$$\bar{p} = \frac{1}{k}\sum_{i=1}^{k} p_i \tag{3.1}$$

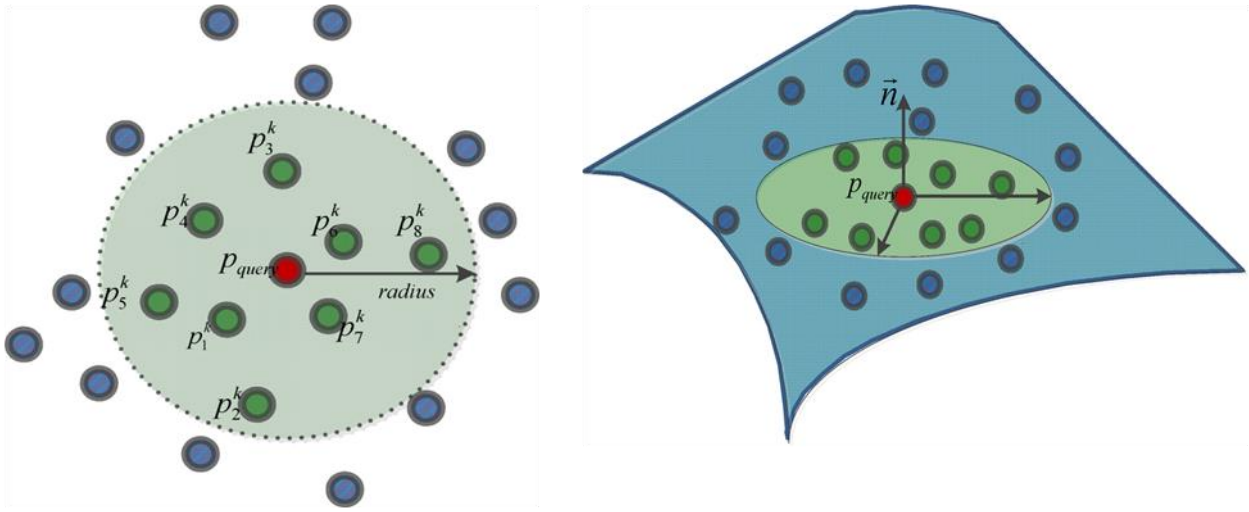


Figure 3. 1 Normal calculation with k nearest neighbor. Left shows red query point's k nearest neighbor points (green points) which are found within Euclidian distance R_{query}. right shows the query point's normal $\vec{n}$ is calculated from the covariance matrix of k nearest neighbor points.

Plane fitting is done by least square methodology i.e., which make points to plane distance square e smallest. $e = \sum_{i=1}^{k} \left(p_i^T \vec{n} - d \right)^2$. k is the number of the considered neighbor points. $||\vec{n}||=1$, $\vec{n}$ is plane normal given by the smallest eigenvector of the covariance matrix M created from the nearest points of the query points. A normal vector projection image is shown in Figure 3.3.

Normal angle threshold T_A and angle between normal of a point and region average normal is used to determine whether the point complies with the criterion or not. The angle is represented as

$$\cos^{-1}\left(\vec{n}_i^T \cdot \vec{n}_{region}^T \right) \tag{3.2}$$

where $\vec{n}_i^T$ and $\vec{n}_{region}$ are query point's normal and already-growed region's normal. The query neighboring point i will add to region relying on criterion

$$\cos^{-1}\left(\vec{n}_i^T \cdot \vec{n}_{region}^T \right) < T_A . \tag{3.3}$$

The second criterion provided by point cloud is curvature. Curvature of points within a plane is usually close to zero. While at sharp edges and corners, the curvauture values are very large. There are several definitions of the point cloud curvature calculation method in point clouds. These include principle curvature κ_1 and κ_2 which are eigenvalues of the shape operator at the query points; mean curvature, $H=(\kappa_1+\kappa_2)/2$, and Gaussian curvature $K=\kappa_1\kappa_2$, where the shape operator is defined as the the negative derivative $S(v) = -DvN$ with N being the normal vector. For point clouds, we tested several calculations but found the following sum of sine angle of the query point and neighboring points is the best one (Equation 3.4). This curvature is regarded as κ_1.

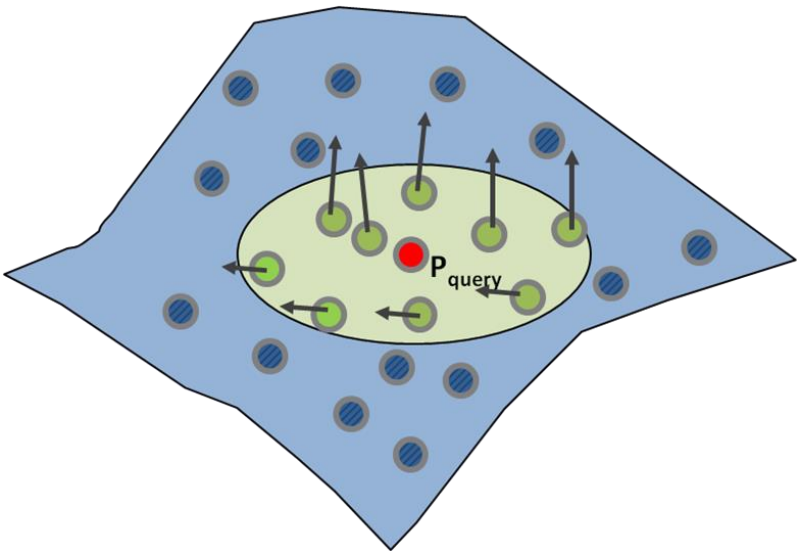


Figure 3. 2 Computation of the curvature of a query point. The curvature is basically local normal variance; hence, the normalized query normal cross product with neighbor points' normal, which is equivalent to sin(α) with α as the angle between normals. Sum projected norm give in the above equation will give an estimation of curvature. Points in plane has small α. Thus small curvature. Edges points has big angles α with neighbor points. thus big curvature.

$$\sin angles = \begin{bmatrix} norm\left(n_{query} \otimes n_1\right)' \\ norm\left(n_{query} \otimes n_2\right)' \\ norm\left(n_{query} \otimes n_3\right)' \\ \cdots \\ norm\left(n_{query} \otimes n_k\right)' \end{bmatrix} \tag{3.4}$$

As we can see from right graph of Figure 3.3, edges have high curvature values. Thus, a threshold can be used to stop the growing.

$$\left|\kappa_{query} - \kappa_{average}\right| < T_k \tag{3.5}$$

$k_{average}$ is the average curvature of points those are aready in the region. T_k is the threshold.

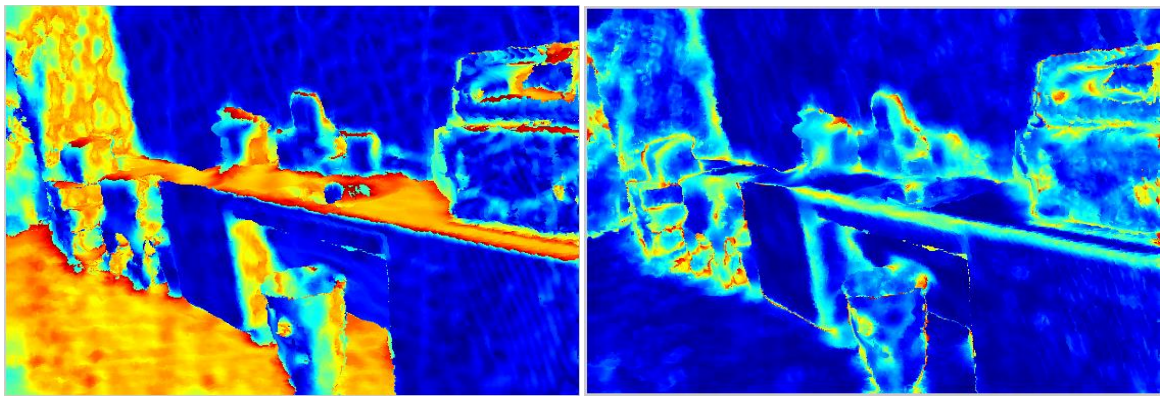


Figure 3. 3 Left: The angle between normal vector and reference vector $(1,1,1)^T/\sqrt{3}$ is calculated to visualize the normal vector in 2D. we see that ground, table top surface has a similar color, thus similar normal vectors. Front surface of table and right wall has the similar blue color. In the curvature image on the right, we can see that the planar region has very low values (blue), and edges have high curvature values.

The region growing is done in the 8-neighbor region of a pixel. The following illustruates the region growing process.

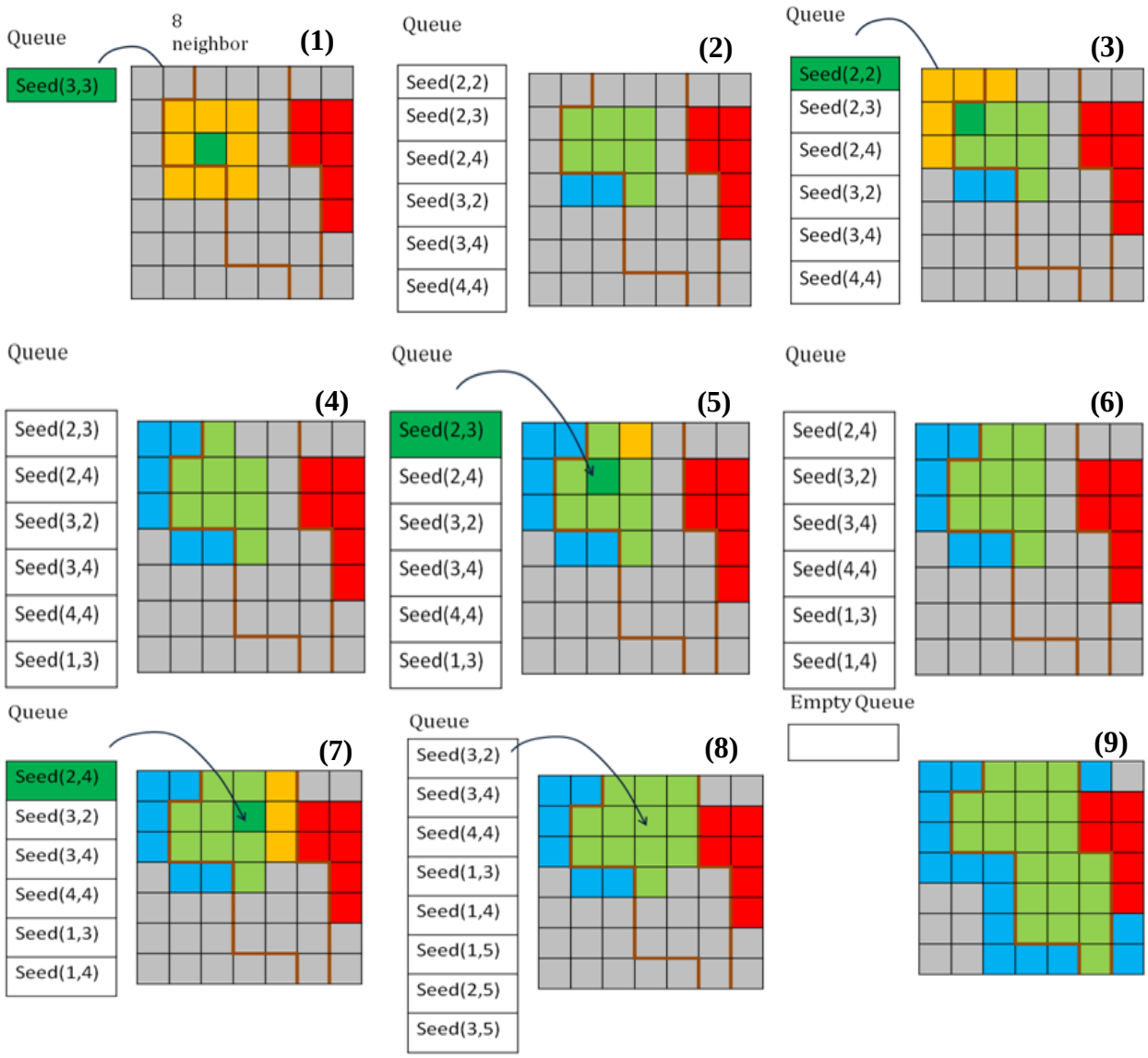


Figure 3. 4 The region grows from (1) to (9). Some iterative steps are not shown. Eight neighbors in yellow that are not labeled in (1). Red are other labels. Green is the seed position, and its position is stored in queue. Determine if any one of the 8 neighbors in yellow belong to the region by comparing color distance, neighbor point to plane (light green region) distance, and angle between neighbor points normal vector and average normal vector (light green region) and curvature difference between neighbor and seed points. Use the new seeds as they continue

to grow. In the last step, there is no seed. The growing stops. The light green region is the newly created region

The region growing process is given in Table 3.1.

Table 3. 1 The region growing

Step 1. Input point cloud P, point clouds' normal N, point clouds' Gaussian curvature C, Lab color image imLab, and thresholds create a label array for labeling and empty label image R.

Step 2. Get a seed from seed list, and put the seed into the newly created queue.

Step 3. Growing from this seed While the queue is not empty, do the following

1) If the seed's pixel position (x,y) of R is not labeled (not visited). Label this seed's pixel by a label, R(x,y)=Ln,

2) Delete the seed from the queue.

3) Get 3D points of P whose corresponding R has been labeled to Ln and calculate centroid of point clouds (X,Y,Z). Select normal vectors of N whose corresponding R has been labeled to Ln and calculate average normal vector; then, normalize normal vectors (Nx, Ny, and Nz). Calculate centroid to normal projection distance Dr. Nx * X + Ny*Y+ Nz*Z + Dr=0 is the plane equation of current region labeled Ln.

4) Visit 8-neighbors of seed. (i=-1,0,1 j=-1,0,1; without i=0,j=0) if the following conditions are met:

If Euclidian distance between neighbor's ab of Lab and seed's ab of Lab is $< T_C$.

If neighbor's point to plane (Nx, Ny, Nz, and Dr) distance is $< T_d$

If angle between neighbor's normal vector and (Nx ,Ny, and Nz) is $< T_A$.

If curvature difference between neighbor and seed is $< T_k$.

Label the neighbor R(x+i,y+j) = Ln and insert neighbor index (x+I, y+j) into queue as new seed.

5 Continue Steps 1–4 until there is no seed in the queue.

Step 4. Repeat Step 2, until there is no seed in the seed list.

Step 5. Fill holes.

The M seeds form N region. N is equal to or smaller than M because some seeds may belong to the same region if region is exclusive; hence, if one region cannot grow to another region, the other region cannot grow into this region either. Second, color threshold and curvature threshold are loose because 1) same plane may have differentcolor, 2) the noisy point clouds do not give

accurate curvature estimation. 3), points to regional plane distance threshold is small, so that the region growing does not grow into different planar regions.

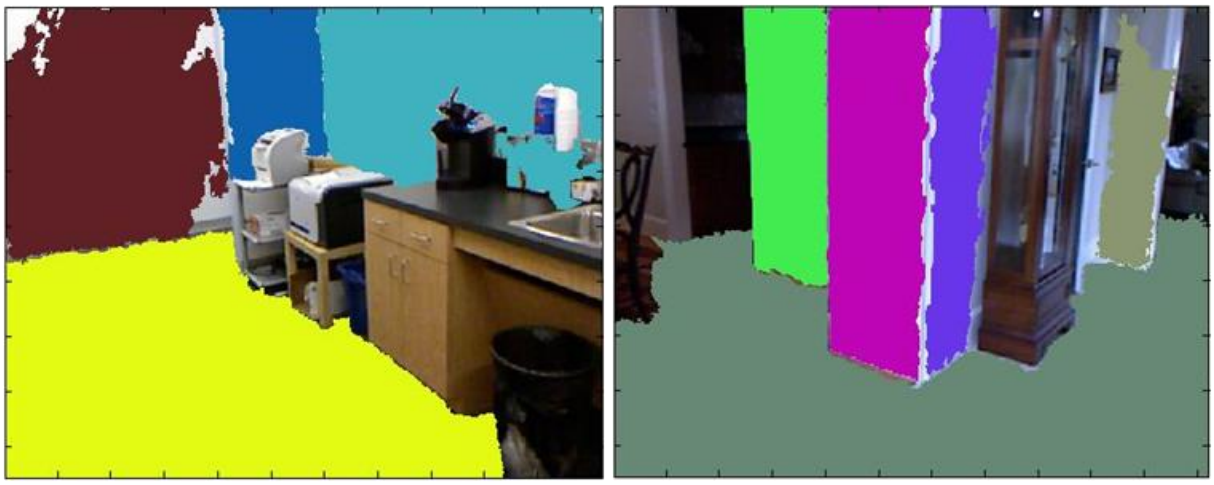


Figure 3.5 The planar region growing exemplary result.

3.3 Region merging using regional adjacency matrix

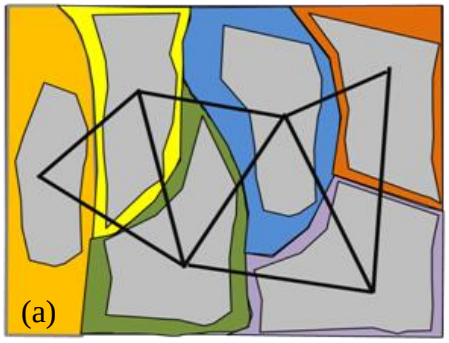
A complete plane (region) is supposed to be segmented into one region, but sometimes a plane can be segmented into multiple regions. Figure 3.5 shows an example. Since these regions belong to one plane, they should be merged together. This section is about region merging. Before the region merging, we need to justify which regions should be merged. Planes that can be merged together satisfy the following conditions: First, they are adjacent neighbors. Second, regional planes are parallel or angled if space between adjacent regional planes is smaller than a threshold, and the distance from a plane's centroid to another plane is smaller than a threshold. These two conditions are also essential steps of region merging. So, first we need to find neighboring regions. A method based on regional adjacency matrix is used to find neighbors [32, 33]. Second, planes are fitted on every region, and two predefined thresholds are used to determine whether neighboring planes can be merged or not. The first threshold is neighboring

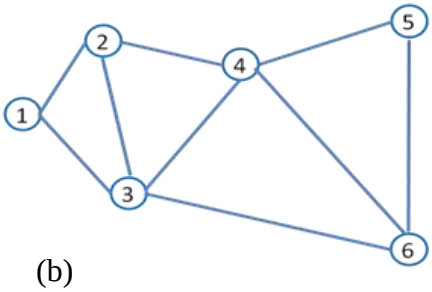
planes' angle, and the second threshold is a region's centroid to another plane's distance. If merging, the neighbor's plane label will change to the label of another region.

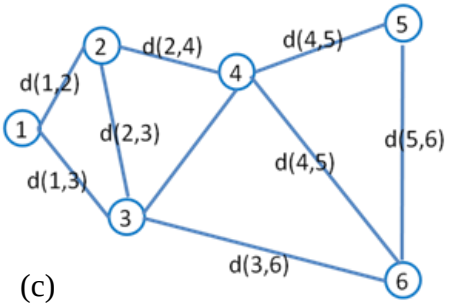
Regional adjacency matrix is a means to tell which regions are neighbors of a querying region. A graph is formed within the regions. Regions are represented as nodes. The connection relationship among these nodes forms the adjacency matrix. Specifically, the adjacency matrix of graph G with n nodes is the n × n matrix where the non-diagonal elements A_{ij} represents (0,1), i.e., 1, where an edge exists from node i to node j and 0, where there are no edges from i to j, and diagonal elements are zeros.

From another side, the region growing method generates regions that are exclusive and isolated islands. Thus the Graph method can be used where each node is associated with a region centroid. Edges are linking centroids. Weights are the distance between centroids. Figure 3.6 illustrates a regional adjacency matrix formation.

The adjacency matrix is fundamental to the region merging procedure [33]. The following illustrations explains the method. A processing example is shown in Figure 3.7. Figure 3.7a is the region growing planar segmentation input. We can see two labeled regions are transposed to a single region. Figure 3.7b is the regional adjacency graph, which indicates adjacency relations among regions. Neighboring regions comply with the conditions where 1) the planar angle is small enough, 2) the centroid of one region to another plane distance is small enough that it should be merged. Figure 3.7c shows the merged regions.







(d)

	1	2	3	4	5	6
1	0	1	1	0	0	0
2	1	0	1	1	0	0
3	1	1	0	1	0	0
4	0	1	1	0	1	1
5	0	0	0	1	0	1
6	0	0	1	1	1	0

Figure 3.6 The formation of a region growing matrix. Figure 3.6a shows region growing segmentation (gray patches) as the input, and the distance image among the regions are computed. Then, the watershed is applied to the distance image to segment the image into complete segmentations (color regions). Corresponding centroids of gray regions are used as the nodes. A graph is formed as shown in Figure 3.6b. The distance between connected centroids is associated with each edge as shown in Figure 3.6c. Finally, Figure 3.6d shows the adjacency matrix.

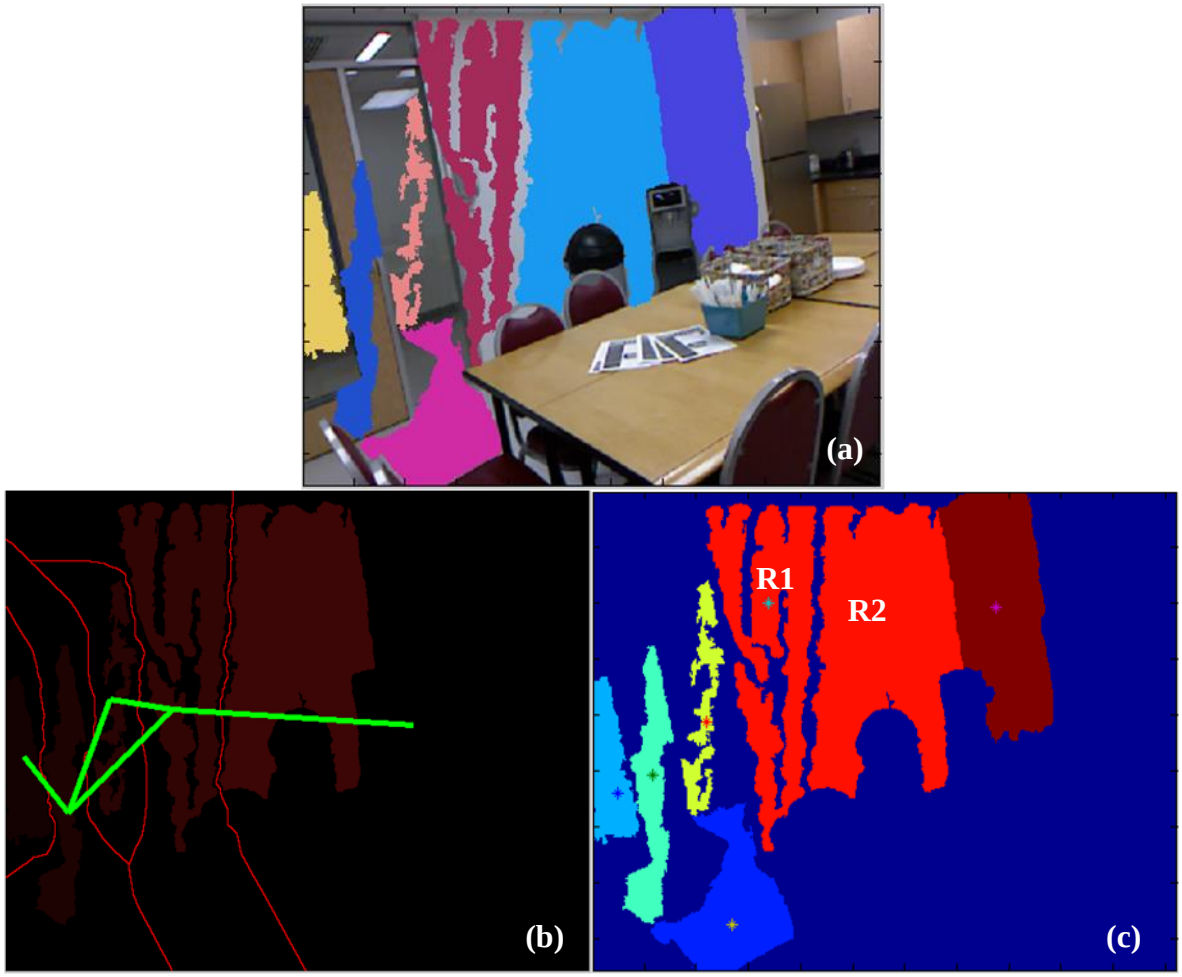


Figure 3.7 An example of region merging. In (a), the input is region growing results. In (b) an adjacency graph is built upon the regions. Neighboring regions are readily obtained from the graph. (c) Neighboring regions that comply with the two conditions are merged into one region. R1 and R2 are merged regions.

From the adjacency matrix we can not only do region merging, but we are also ready to get the regions' relations, i.e., which region is which region's neighbor. With these neighboring region locations, we can also obtain each plane's alignment with the world axis. Combining these two priors, we know the adjacent regions' relations and whether they are parallel or orthogonal. This knowledge was used in Section 4 of 3D modeling.

3.4 Planar region growing results and comparison with RANSAC plant fitting

Besides our region growing method, another robust planar region segmentation method uses RANSAC. For comparison of their segmentation results, we used the RANSAC region segmentation program developed by Schnabel et al. [35]. [35] has a link which takes the reader directly to a detailed account of the Schnabel et al. program development.

Same images are tested on both segmentation programs. The results are shown in Figure 3.8.

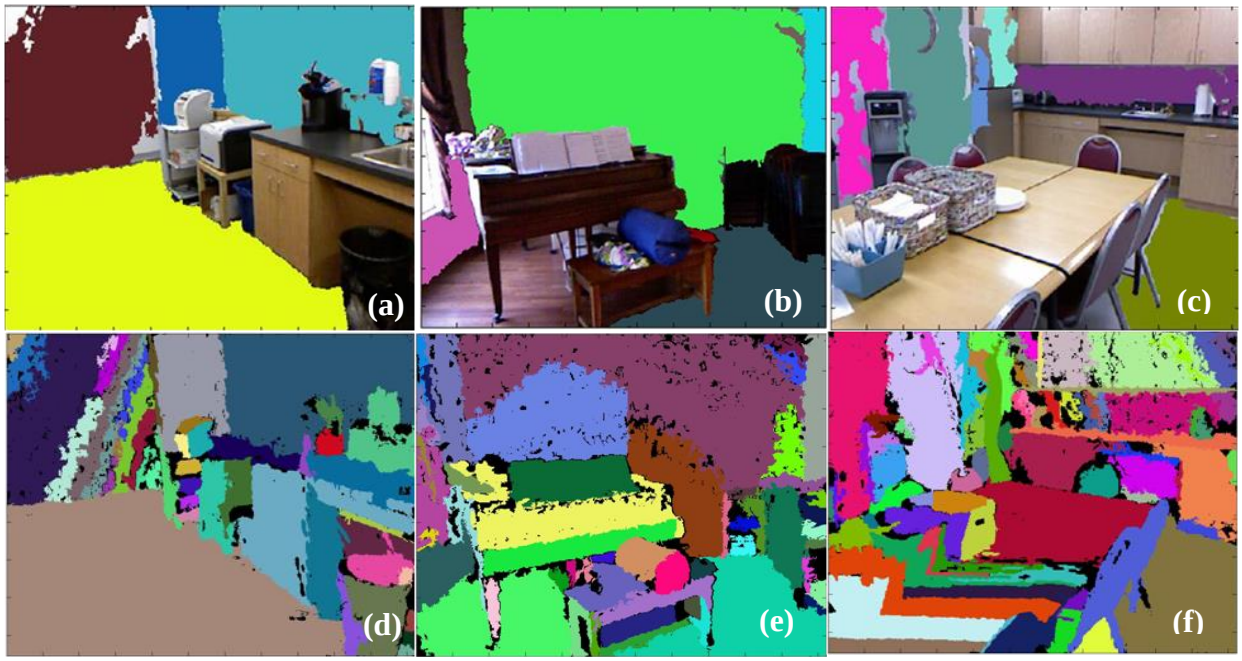


Figure 3.8: RANSAC segmentation results. In this part, a ,b, and c are segmentation results of region growing. The RANSAC segmentation of the same images are shown as d, e, and f. Images from this same Figure 3.8g–r are shown on the next page.

Figure 3. 8 continued: Planar region growing segmentation and RANSAC segmentation comparison. In figures g, h, i, m, n, and o, we can see region growing segmentation results.Corresponding RANSAC segmentation results are shown in figures j, k, i, p, q and r.

We can see from these image processing results that region growing gives relatively large regions and fewer regions based on the seed numbers and coverage over the image. RANSAC tends to produce trivial fragments especially at far distances where the points' measurement are noisy. Another different characteristic is that region growing tends to generate smooth, isolated islands, however RANSAC usually produces regions comprised of several subregions. Crossing among regions in RANSAC are ubiquitous. This is caused by the properties of RANSAC's plane fitting algorithm. Points belonging to the same plane are mathematically obtained no matter where they are located. However, RANSAC is much faster than the region growing algorithm. So, RANSAC is preferred in the following for its ability to perform quick modeling. Region growing modeling can be used to obtain higher accuracy of details.

CHAPTER 4. 3D MODELING

4.1 The method of modeling

The indoor scene is assumed to be Manhattan world. Thus, it concludes that any adjacent two planes are orthogonal in the real world. However, due to the projective imaging, some planes are blocked and, thus, they do not show in the image. The blocked planes' left and right neighboring planes are shown in the projective plane and become neighbors. The corresponding real world planes of these "neighboring planes" (not neighbors in the real world) are either orthogonal or parallel. For the modeling, the procedure is divided into four major steps: 1) initial outlier plane is cleaned and floor detection is carried out to clean the outliers and find floor with the aid of major orientation obtained from last chapter's planes or world coordinates from VPs. Major orientation and world coordinates are equivalent. 2) Automatic wall detection is based on the visibility of regional points projected on the floor plane. 3) Wall boundary points are detected. 4) 3D reconstruction uses the detected walls, floor, and boundary points through a method called domain partition. Details of the implementation of these steps will be described in the flowing sections.

4.1.1 Step 1: Initial outlier plane cleaning and floor detection

The outlier planar regions refer to those that first have a very small area and to those that are not "aligned" with any of the major orientations. Area of regions is defined as the points that belong to the region. Total area is the total number of the points. A threshold is first predetermined by observing the segmentation to remove regions within small area. The threshold was set to 0.02, which means the region was removed if its number of pixels was less than 2% of the total pixels. Alignment of the planar region was done to compare each region's normal with

the major orientation and determine which orientation had the smallest compatible angle. Also an angle threshold was set to insure that the smallest angle was small enough; otherwise, the region was discarded as outlier.

Each planar region was fitted with an optimized least square plane equation through the RANSAC procedure. The plane had a normal n and distance d from the origin point (0,0,0) which is actually the camera center. The following equation shows how the threshold is given. $\min\left(n \bullet N_x, n \bullet N_y, n \bullet N_z\right) < \gamma_{threshold}$, where, (Nx, Ny, and Nz) are the three unit major orientations. γ is the angle threshold angle.

After two rounds of cleaning, relatively "good" regions remained and were classified according to their alignment. The following Figure 4.1 shows a processing example. Figure 4.1a is the region labeled according to RANSAC planar fitting on point clouds. Each region has a unique labeling number and is colored with a unique color. Figure 4.1b shows the changes after two thresholding procedures and an alignment.

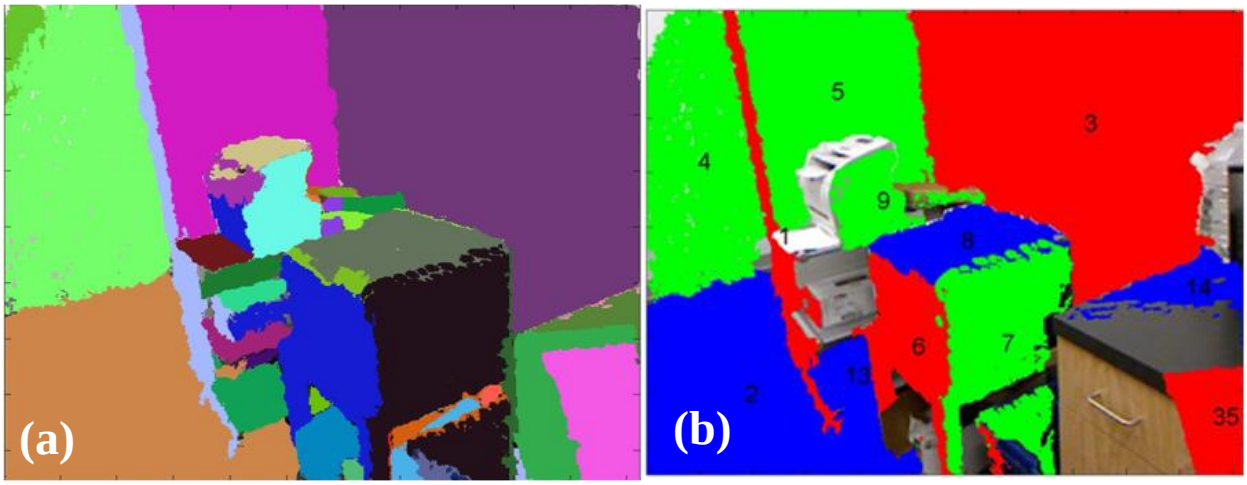


Figure 4. 1. Here, (a) shows RANSAC segmentation results. In (b), we see the image after it was processed by area, alignment angle thresholding and alignment assignment.

Small regions are removed. Planar regions are unaligned with any major orientation by threshold are removed. The remaining regions are labeled with red, green and blue colors if they are aligned with N_x, N_y, and N_z, respectively. These will be passed down for further processing. Red and green regions are parallel to wall regions. The blue regions are parallel to the floor plane. The region located at the lowest level is the floor plane. Before finding the floor plane, the point cloud is rotated with a TBN matrix, which is an orthonormal matrix obtained from major orientations. TBN = (N_x, N_y, and N_z)

$$TBN = \begin{pmatrix} T_x & T_y & T_z \\ B_x & B_y & B_z \\ N_x & N_y & N_z \end{pmatrix} = \begin{pmatrix} N_x' \\ N_y' \\ N_z' \end{pmatrix} ,$$

The rotation of point cloud with TBN matrix is given as $(x', y', z') = (x, y, z) \times (TBN)'$, where ` is the matrix transpose operator. This rotation will make each blue region basically have a constant height z' value making it parallel to the screen, while red and green regions will be perpendicular to the screen.

With rotated point clouds, the floor is defined as having the z coordinate extremum of the blue region centroid (z-aligned regions) after rotating the point cloud to lie parallel to the blue planes. The object is to obtain a centroid of z coordinate for each blue region and to find regions of extremum centroids, i.e., with very low height values. Furthermore, if some of the extremum values are very close, the areas are further compared to select region with bigger areas. For the extremum comparison, a threshold |z'i-z'j |< 0.1 is given to determine whether two regions are close or not.

Figure 4.2 shows an example of finding a floor. Figure 4.2 shows an example with floor detection. Blue regions 8, 2, 13, and 14 are aligned with the N_z axis of the major orientation, which

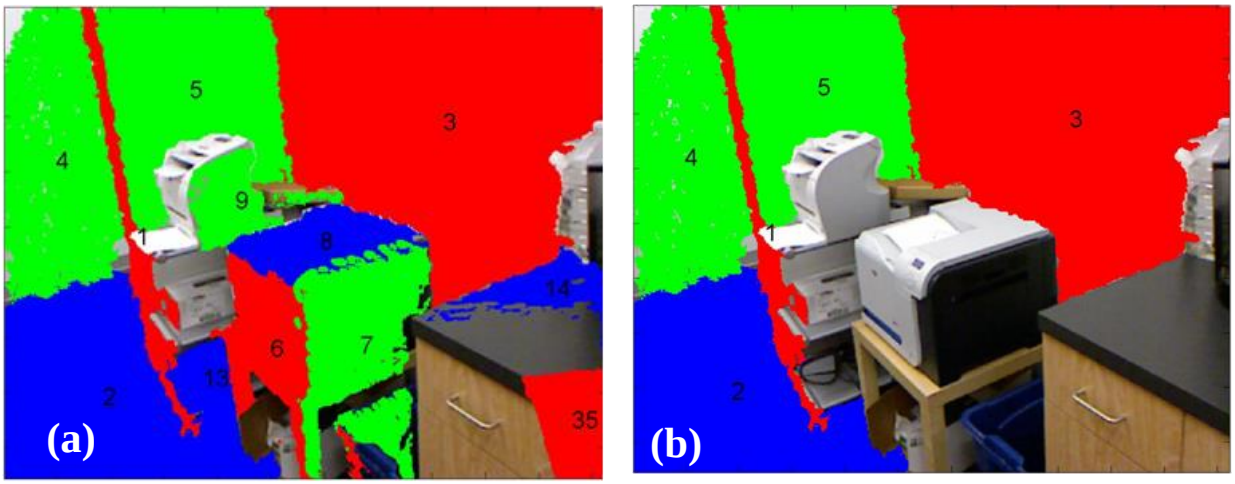


Figure 4. 2 In (a), we see the regions after the initial cleaning and alignment. In (b), the floor is detected and reserved, which is shown as Blue Region 2.

indicates they are potential floor candidate. With rotation, each region basically has constant z' values of rotated point coordinates (x', y', z'). The centroid, i.e., mean value of z' of each region is calculated and compared, $z_2^{centroid} \approx z_{12}^{centroid} < z_9^{centroid}$. Region 2 and Region 13 are close, but Region 2 has a bigger area. Thus, Region 2 is detected as the floor.

With this processing of alignment labeling, initial cleaning, and point cloud rotation with TBN matrix and floor detection, the data can be passed on for use in the second step of automatic wall detection.

4.1.2 Step 2: Automatic wall detection based on visibility of regional points.

In order to get a correct automatic pick-up of boundary points, we should have a correct detection of wall planes. Some preprocessing such as the projection of red and green regions to the floor plane and region edge detection are necessary to aid in wall detection. With the aid of preprocessing, correct wall region detection is performed, which removes the outlier regions and reserves the right wall regions. The methods for these steps are given as follows:

1) After processing of Step 1, the initially cleaned points are transformed to be parallel to the blue floor plane as detected from Step 1. The points will be represented by (x', y', and z'). Red and green regions are potential wall candidates. Wall detection is performed from the projection of red and green regions to floor plane, We first project the red and green regions to floor plane by setting a constant z' value on red and green regions' point. However, the constant z' is not important, since we processed on a 2D projection plane. The z' values are simply removed, and just (x' and y') are reserved to be the projection. Figure 4.3 shows the 2D projection of red and green regions after rotation by simply removing the third coordinate z'. Each segment in Figure 4.3b has a corresponding region in Figure 4.3a. Each region becomes line clusters. RANSAC line fitting is applied on each line cluster to get the best line fitting. After finding the best line, the inliers must be projected to the line so the projections can be used to find the two edge points, which become the initial boundary point of that segment. These points are shown as blue dots located at end of the line clusters. Segments are used to indicate the same thing, because some of them are outliers and are not from the wall segment. A segment is used really because some of them are only part of the whole wall segment. From the inliers of each

RANSAC line, we can also obtain center points by averaging inliers' coordinates. These points are shown as magenta dots located in the center of the line clusters (Figure 4.4). Yellow points are the intersections of all the RANSAC lines. They are not useful here and should be ignored.

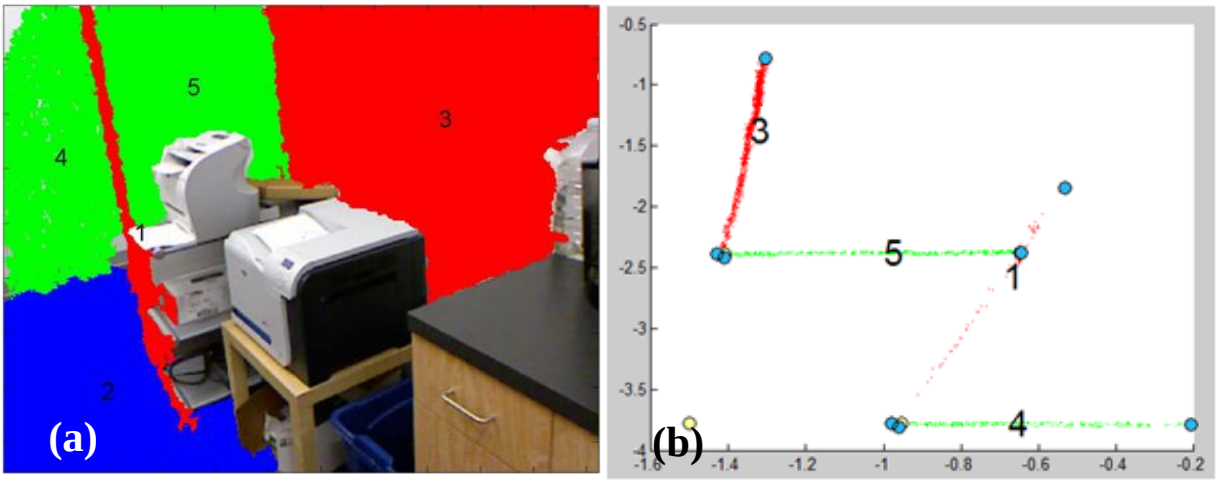


Figure 4. 3 shows the projection of the red and green regions by simply removing z' values of rotated points. In (a), the initial processing is with Step 1. In (b), we see the projection of the red and green regions. Labels are displayed to match regions in (a) and (b). Blue points are the obtained edge points of each segment by RANSAC line fitting. The yellow dots are the intersection points among the line segments.

2) Visibility detection and automatic wall detection. Figure 4.4 is the example of processing. Remember the rotation of the point cloud will not change the camera center, whose coordinate (0,0,0) is preserved. Thus the (0,0) point at the projection plane is the camera center projection. It is shown as a pure blue origin point O at (0,0) coordinate after

projection to floor plane. The visibility is detected with respect to (0,0). Two cyan lines with origin point O form the coordinate, which is referred to as the cyan-O coordinate. Connecting the center point of each RANSAC line with the origin O, we can define the angle of each line segment. After sorting the angle from small to big, I was able to determine the sequence of these line clusters when counted counterclockwise in the cyan-O coordinate. For instance, the sequence is 3, 11, 9, 4, 1, 7, 8. The following processing will use this sequence when iterating all the segments.

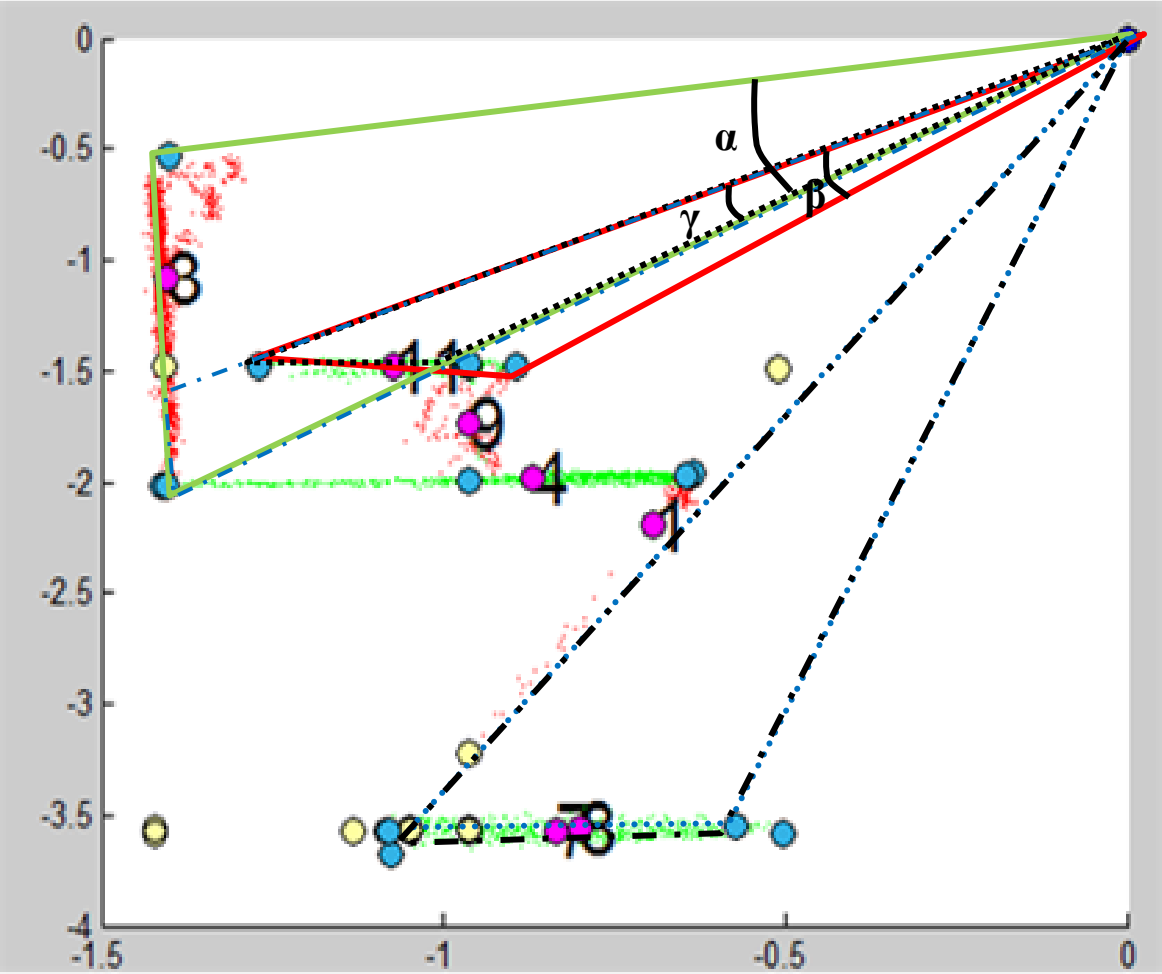


Figure 4. 4 The visibility detection based on the fan extending from each region. The blue dots define the edges of the fans. The magenta dots are the centroids of the segments.

By connecting blue edge points of each segment with origin O, we can define the maximum and minimum span angle of the segment with respect to origin. For each segment, compute the overlapping relation between it and all other segments by comparing the max and min span angle of each segment. For instance, in Segment 3‘s min and max span angles are formed by $\overline{aO}$ and $\overline{bO}$, respectively. Segment 11’s min and max span angle are formed by $\overline{cO}$ and $\overline{dO}$, Segment 3’s span has an overlap with segment 11’s span, which is shown as angle γ. Hoeever, under the overlap angle γ, the triangle is $\overline{cfO}$ for Segment 11, where f is the intersection point of line $\overline{cd}$ and $\overline{bO}$, and the triangle is $\overline{ebO}$ for Segment 3, where point e is the intersection point of line $\overline{ab}$ and $\overline{cO}$.

Each segment will create an overlap relation matrix with all the other segments. If there is overlap, an overlap ratio, for instance for Segment 3 with respect to 11-- γ/α, is defined. Overlap areas such as triangle $\overline{\text{ebO}}$'s area and triangle $\overline{\text{cfO}}$'s area are calculated also.

For each segment with respect to other segment, we maintained these three values: an overlap ratio and two overlapped areas. We then use them to identify whether a segment is an outlier. First, compare te overlap ratio with a threshold. If it is > 0.3, then, compare its area with other one. If its area is bigger, then it is not recognized as an outlier. If its area is smaller, then it is labeled as an outlier once. For instance, if $\overline{\text{ebO}}$'s area > is $\overline{\text{cfO}}$, Segment 3 is not labeled as an outlier. Segment 11 is labeled as outlier once. We then iterate this comparison over all other segments. Each segment may be labeled as outliers several times. For instance, Segment 11 was labeled as an outlier twice by comparing it with Segment 3 and Segment 4.

If two segments have overlap, and two other segments are also overlapping an area that is very close, i.e., 0.95< (first triangle’s area)/ (second triangle’s area) < 1.05, they are

recognized as being able to merge as one segment as shown in the merging of Segment 7 and 8 due to this type of overlapping. The area of triangle $\overline{ghO}$ and the area of triangle $\overline{g'h'O}$ are close enough that they should be merged.

With the visibility calculation, Segments 7 and 8 were merged before the following calculation. If two segments should be merged, the new edges points are updated according to the merged points of these two segments. The visibility calculation is performed again on segments after merging. From Segments 3, 11, 9, 4, 1, and 7, Segments 11, 9, and 1 should be deleted, and Segments 3, 4, and 7 are reserved as being the detected walls. Further, RANSAC lines are recalculated on the wall segments. Edge points are also recalculated.

The following Figure 4.5 shows an example of the preceding processing. 4.5(a) is the result of Step 1 processing. (b) is the projection of red and green regions to floor. Visibility detection shows that Segments 7 and 8 should be merged. (c) shows the merging results. Visibility is utilized again to remove outlier regions 11, 9, and 1. (d) shows the final detected wall segments. The following two figures show two more representatives of automatic wall detection examples.

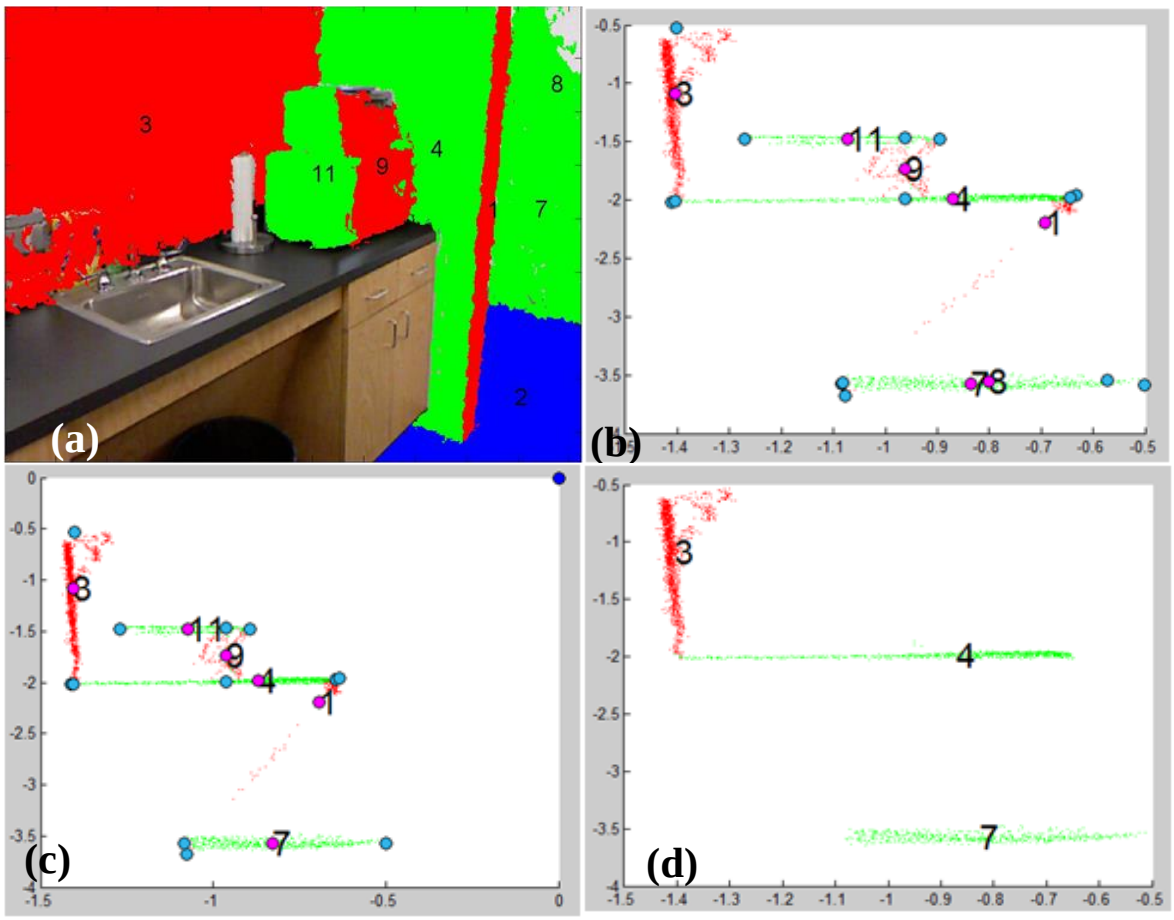


Figure 4. 5 Automatic wall detection through visibility. (a) shows the preprocessing of point cloud using the Step 1 method. (b) projects red and green regions. Visibility calculation is first performed, which indicates Segments 7 and 8 should be merged. (c) shows the merged segments. Visibility is calculated again to remove outliers based on new segments after the merging processing. (d) shows the wall detection by removing the outlier segments with the aid of visibility detection.

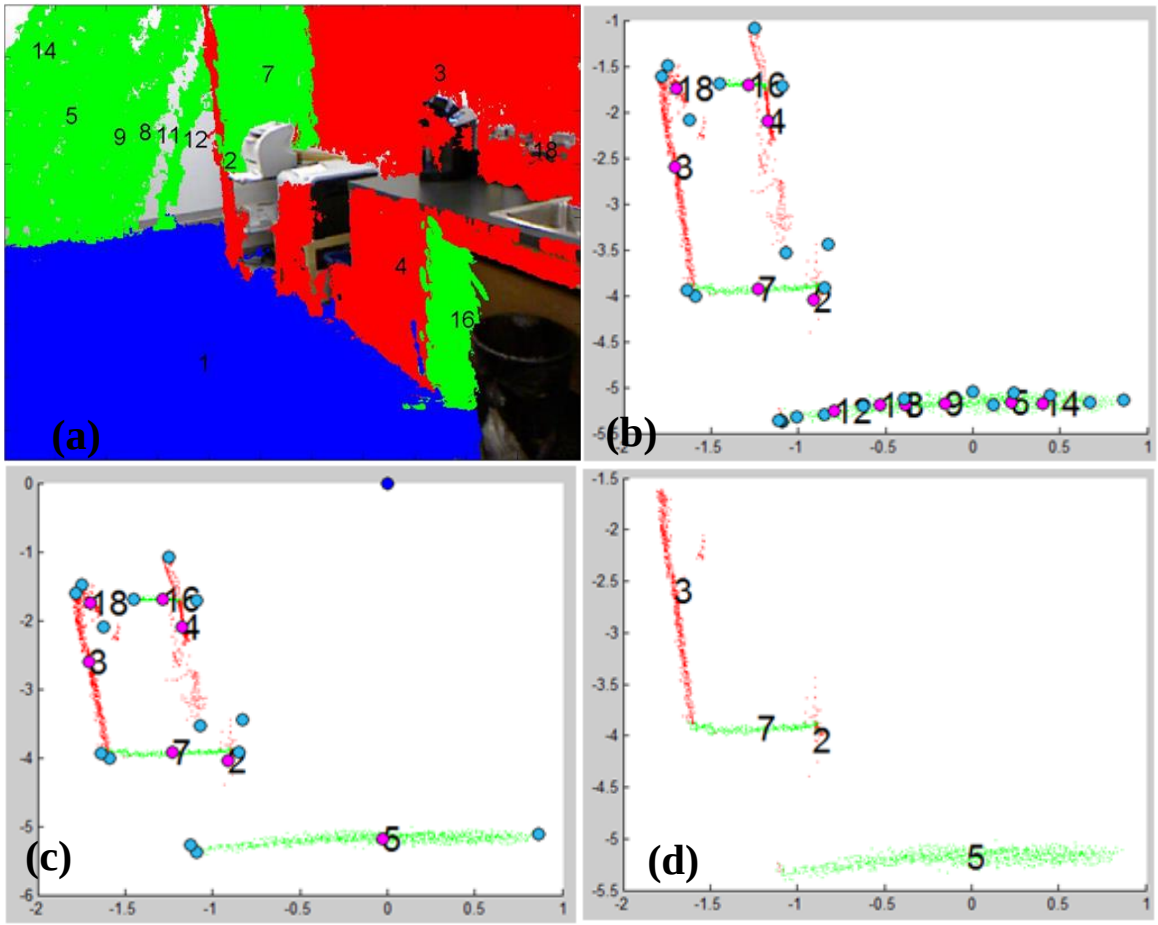


Figure 4. 6 This figure is similar to Figure 4.5, but with more segments that should be merged. a) shows the preprocessing of point cloud with using the Step 1 method. (b) projects red and green regions. Visibility calculation is first performed to detect visibility, which indicates Segments 5, 9, 8, 11, 12 and 14 should be merged. (c) shows the merged segments. Visibility is calculated again to remove outliers based on new segments after merging process. (d) shows the wall detection by removing the outlier Segments 18, 16, and 4 with the aid of visibility detection.

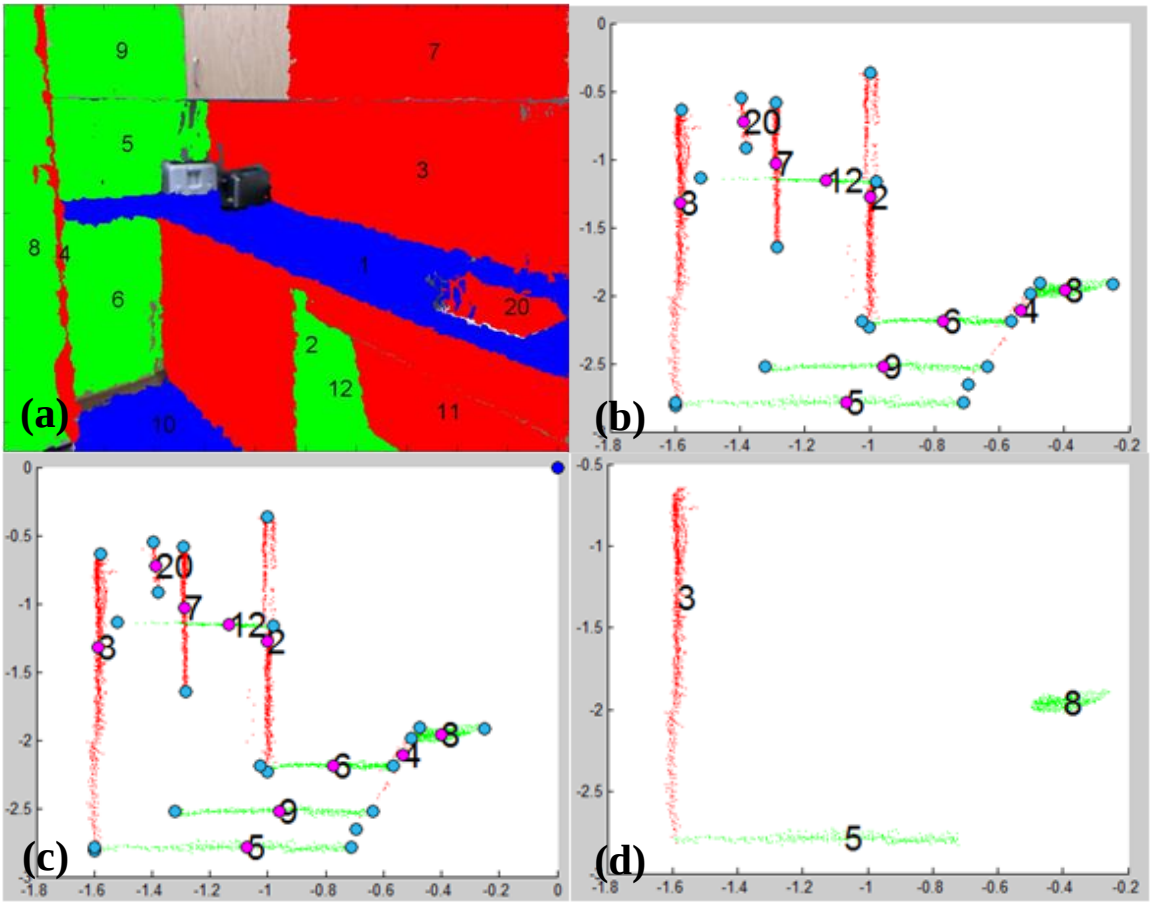


Figure 4. 7 This is similar to Figure 4.5, but with more outlier segments that should be removed. a) shows preprocessing of point cloud using the Step 1 method. (b) projects the red and green regions. Visibility calculation is first performed, which indicates no segments should be merged. (c) Merged segments are identical to (b) because no segments were merged. Visibility is calculated again to remove outliers based on new segments after the merging processing. (d) shows the wall detection by removing the outlier segments 20, 7, 12, 2, 6, 9, and 4 with the aid of visibility detection.

4.1.3. Step 3: Automatic boundary point detection

This section discusses how to determine boundary point detection. After getting clean data as shown in (d) of Figures 4.5–4.7, we could detect the boundary points. The following is

boundary point calculation method: From the segments' sequence, i.e., the sequence obtained by comparing the centroid angle with respect to origin (0,0), we know each segment's front and rear neighbor. For a segment, if it has no left neighbor, its right blue edge points are saved as its left boundary wall points. If it has no right side neighbor, its left side blue edge point is saved as its right boundary wall points. If it has a left and right neighbor, its boundary wall points are obtained by the following further processing: (1) If its neighbor has a different color, indicating its neighbor is orthogonal to it, this boundary point is obtained as the intersection point of their RANSAC lines. (2) If its neighbor has same color, indicating its neighbor is parallel to it, this boundary point is its blue edge point because we don't have a RANSAC line intersection point.

After determining the two boundary points of each wall segment, we project them to the floor plane. The intersection points are the wall boundary intersection points with the floor plane.

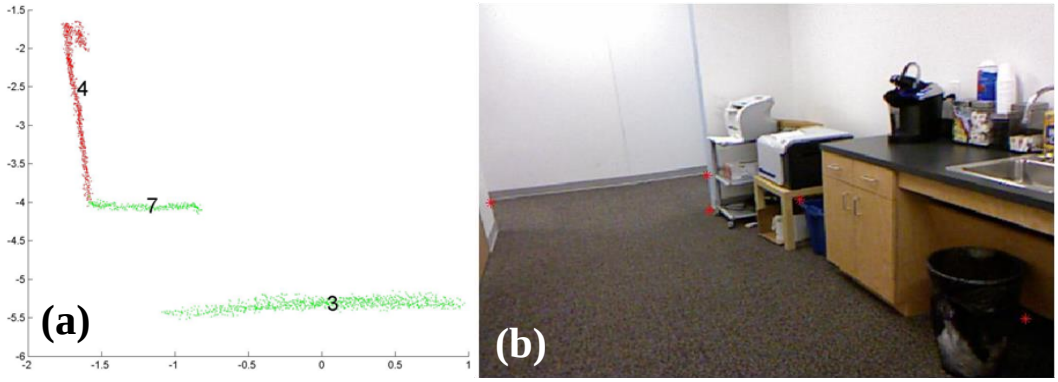


Figure 4. 8. The wall automatic boundary point detection. Boundary points are first detected at the projection plane. Their projection at floor plane gives the boundary point intersection with floor plane. (b) shows the obtained boundary points shown as red star dots.

4.1.4 Step 4: 3D modeling by extruding from automatic boundary point detection

From above work, we have been able to detect the boundary point automatically. In this step, a 3D modeling method by extruding from boundary points is explained. Here, we have a floor region fitted with a plane (n, d), where n is 1×3 normal, d is distance from origin point to plane. The plane equation is represented as $n \odot x - d = 0$, where $\odot$ is the dot product. The detected boundary points are also at floor plane. Each wall has two boundary points, e.g., xl and xr, which are referred to as left and right boundary points. Boundary points with normal n define the wall boundary lines. So, if we can extrude the wall plane enclosed between two boundary lines from the floor, we can get 3D modeling. But it is difficult to grow the wall in 3D space. I finished this with the aid of 2D segmentation.

First: Left boundary 3D line can be defined by n*t+xl, where xl is the detected boundary point, and n*t+xl = x l2, which gives the second boundary point along the same line. Project xl and xl2 extend to 2D camera plane, which defines the projected boundary line in 2D, e.g., line L_left. Similarly, the right boundary 3D line gives a projected boundary line of R_right in 2D. These two boundary lines in 2D defines the left and right limit that the wall can extend to. Similarly, every other wall has two 2D boundary lines created from automatically detected boundary points, which subsequently defines the wall's left and right extension limit. We could call the obtained image from above method *wall domain*.

Next : In each domain, we have two planes defined. One is for the wall, and the second is the floor plane. Every point compares with the wall plane equation. The wall plane normal is given as $n\_wall = (xl-xr) \otimes n./\| xl-xr \|$ where $\otimes$ is the cross product, and ||.|| is norm, while n is normal of floor plane. This means the wall's normal is given as a cross product of floor normal

and a vector is defined in the wall plane. The wall plane is given as n_wall ⊙ (x-xl) = 0 or n_wall ⊙ (x-xl) = 0, where ⊙ is dot product.

The floor plane equation is n⊙ x – d = 0 as given in the beginning. For each pixel in the domain, two 3D points (xfloor, yfloor, and zfloor) and (xwall, ywall, and zwall) are calculated by intersecting the ray line (emitting from camera origin and passing through the pixel) and the two planes. Compare zfloor, and zwall. If zfloor < zwall, (xfloor, yfloor, and zfloor) is reserved, otherwise (xwall, ywall, and zwall) is reserved.

The following Figure 4.9 shows the intermediate results of the processing steps. In Figure 4.9a, RANSAC regions align with major orientation, as the floor has been detected, and some initial cleaning has been done. Figure 4.9b shows the remaining wall projection on the floor plane after automatic wall detection. Intersection points of neighboring line segments or edge points will give the boundary points based on neighboring segment relations. Projecting these boundary points to the floor will give the boundary points on the floor. These points project to the 2D plane and are shown as red points in c and d. The next step is to find a second point along each boundary line of wall along floor normal direction, starting from boundary points on the floor. Project them to the 2D plane. This gives the upward red point along the wall boundary line in Figure 4.9d. Each wall's left boundary points (two red points in 2D, for example marked by circle in 4.9d) define the left limit of the domain in 4.9e as well as the right boundary points (two red points in 2D, for example marked by rectangle in 4.9d) define right limit of the domain in 4.9e as the arrow indicates in 4.9f. In each domain, for each pixel, a ray emitted from origin will intersect with the floor plane and wall plane, i.e., the (x, y, z), where a small z is reserved. Figure 4.9g shows the reconstruction with artificial color for each region. Figure 4.9h shows the reconstruction with color texture.

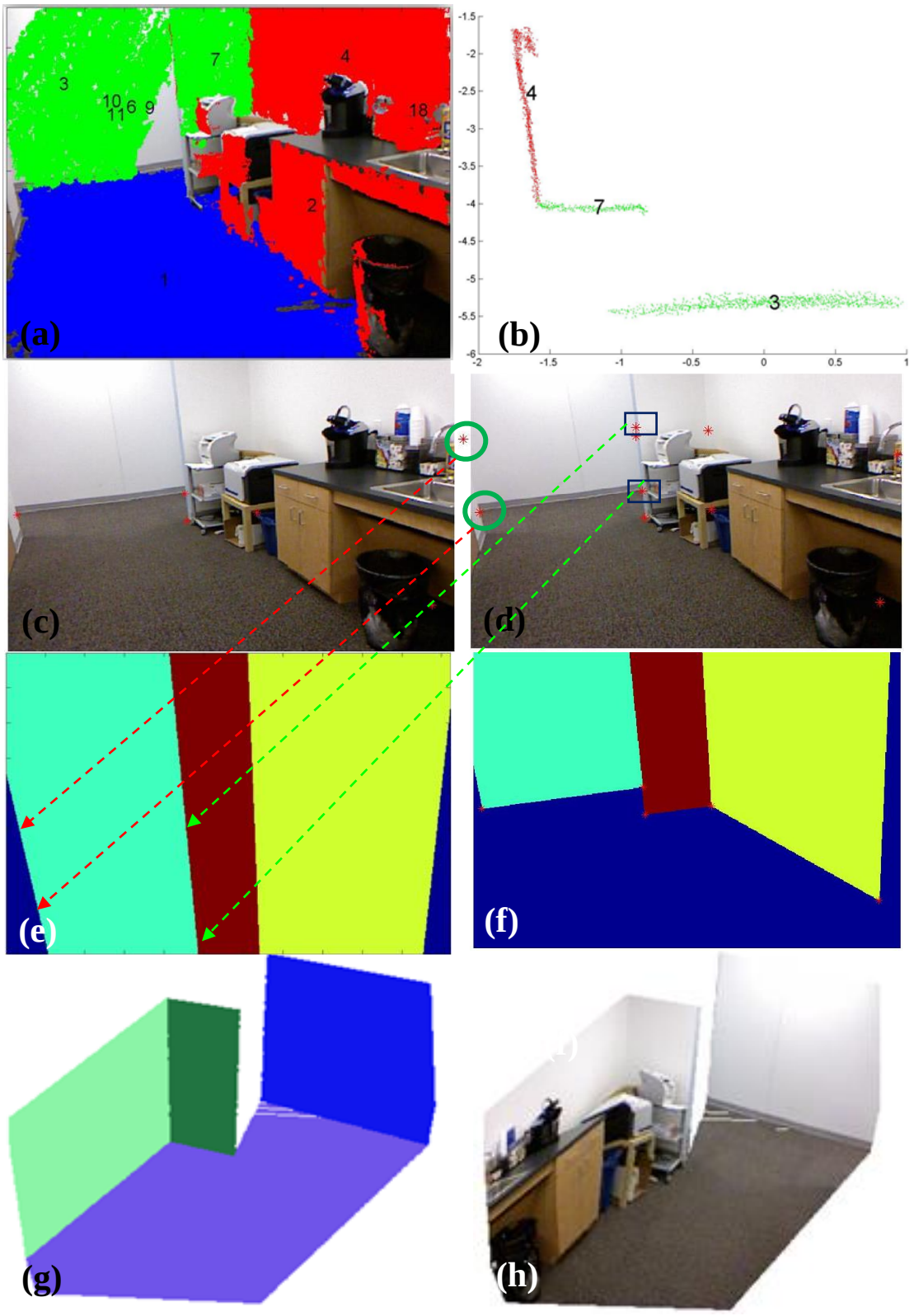
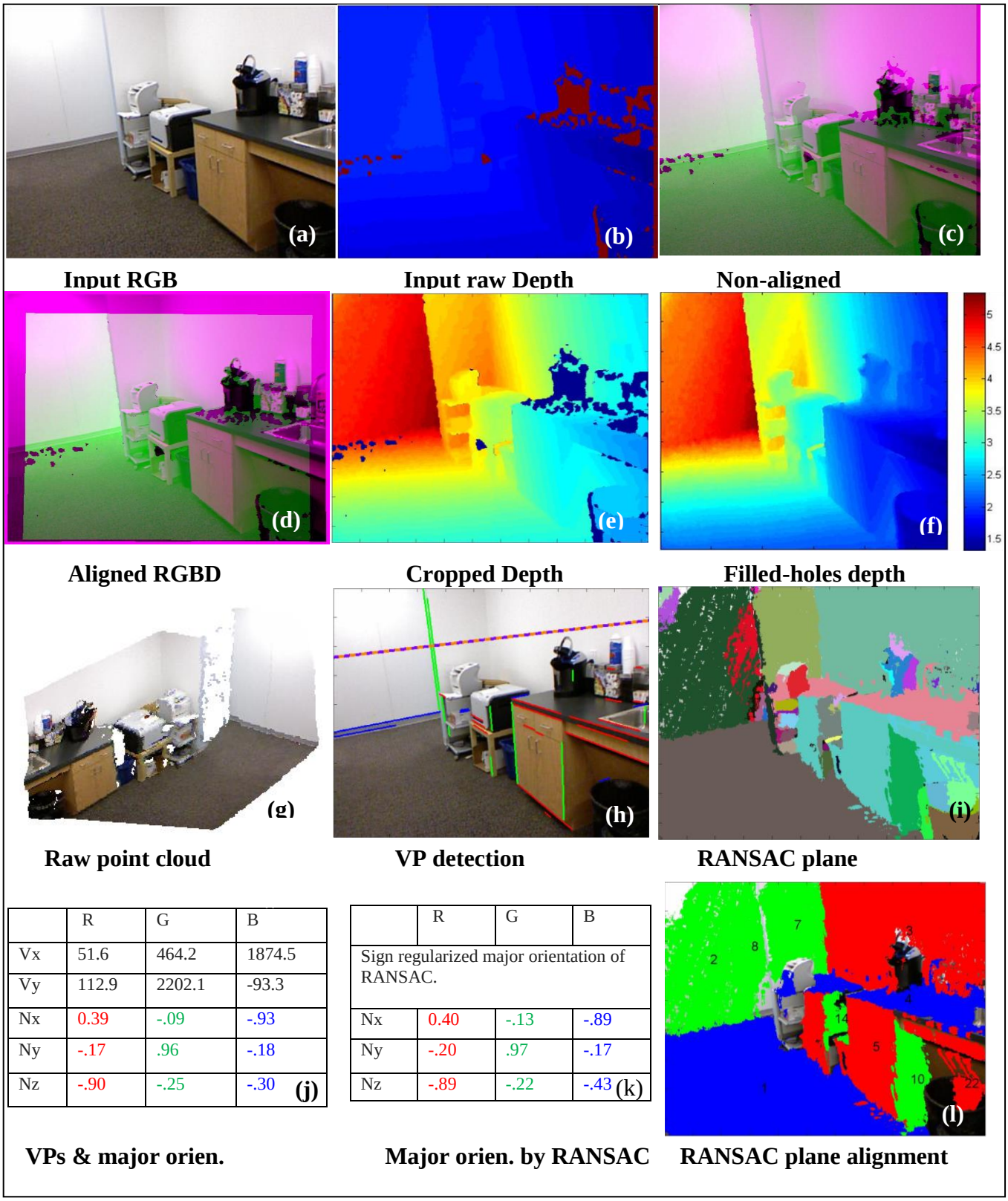


Figure 4. 9 3D modeling. Detailed explanation is in the Section 4.1.4 narrative.

4.2 Modeling pipeline

In this section, a detailed modeling pipeline is summarized and illustrated. One is shown with real data. The other is shown with synthetic data. Figures 4.10 and 4.11 show the modeling pipeline with a representative sample. Figure 4.10a and b are the input rgb and raw depth (D) images respectively. Combining a and b gives c, which shows that RGB and D are not aligned because they are from two sensors with different viewing angles. Figure 4.10d shows the aligned RGB and D image by transforming the D image from its own sensor frame to the RGB sensor frame. Figure 4.10e shows the cropped depth image after alignment. Figure 4.10f shows the depth image after hole filling with bilateral filtering. Figure 4.10g is the raw point clouds obtained by combining the RGB image and D image. Figure 4.10h shows the vanishing point detection results. Figure 4.10i shows the RANSAC planar segmentation results. Figure 4.10j shows the RANSAC image from vanishing points so we can deduce the major orientations of the planes, which is equivalent to the world axis. Figure 4.10k shows the major orientations obtained from the RANSAC plane. Initially, these major orientations had different "+" or "–" signs from that of VPs. After sign regulation, these major orientations are very close to that of VPs which can be seen by comparing Nx, Ny, and Nz. These major orientations can first be used to find the alignment of each RANSAC plane, which is shown in 4.10l. The alignment can subsequently be used to detect the floor, which is shown in 4.10m. Secondly, these major orientations can be used to rotate the point clouds parallel to the floor plane, where the planes that are perpendicular to the floor will be shown as segments 4.10n1. Figure 4.10n2 shows the results of merging segments. As we know, some of the segments belong to the same plane, the merging process is done here in this step. From the merging result, the visibility can be calculated, a visibility voting matrix is obtained, which will be used to determine which segments will be deleted. The

reserved segments are the real walls. Figure 4.10n3 show the automatic wall detection results. From Figure 4.10n3, we can detect the boundary points between adjacent walls depending on the neighboring relations. If two walls are parallel, the edge points of each segment are their corresponding boundary points. If two walls are orthogonal, the intersection points are the boundary points. These obtained boundary points are actually lines perpendicular to the floor. With this prior, the boundary points on the floor plane can be obtained by calculating the projection points from boundary points to floor. Figure 4.10o1 shows the boundary points on the floor. (o2) shows the second boundary points away from floor. Two boundary points define the real boundary lines in 4.10p. Boundary lines define the domain image, which limits the wall's left and right extension range. Figure 4.10q. In each pixel of a domain, comparing the depth obtained from the floor plane and depth obtained from the corresponding wall plane will determine if this pixel belongs to the wall or floor. This processing will give the 2D modeling image in Figure 4.10q. From 4.10q, we can obtain the 3D modeling 4.10r easily. Figure 4.10s shows the 3D modeling with texture.



	R	G	B
Vx	51.6	464.2	1874.5
Vy	112.9	2202.1	-93.3
Nx	0.39	-.09	-.93
Ny	-.17	.96	-.18
Nz	-.90	-.25	-.30

	R	G	B
Sign regularized major orientation of RANSAC.			
Nx	0.40	-.13	-.89
Ny	-.20	.97	-.17
Nz	-.89	-.22	-.43

Figure 4. 10 The modeling pipeline with real data.

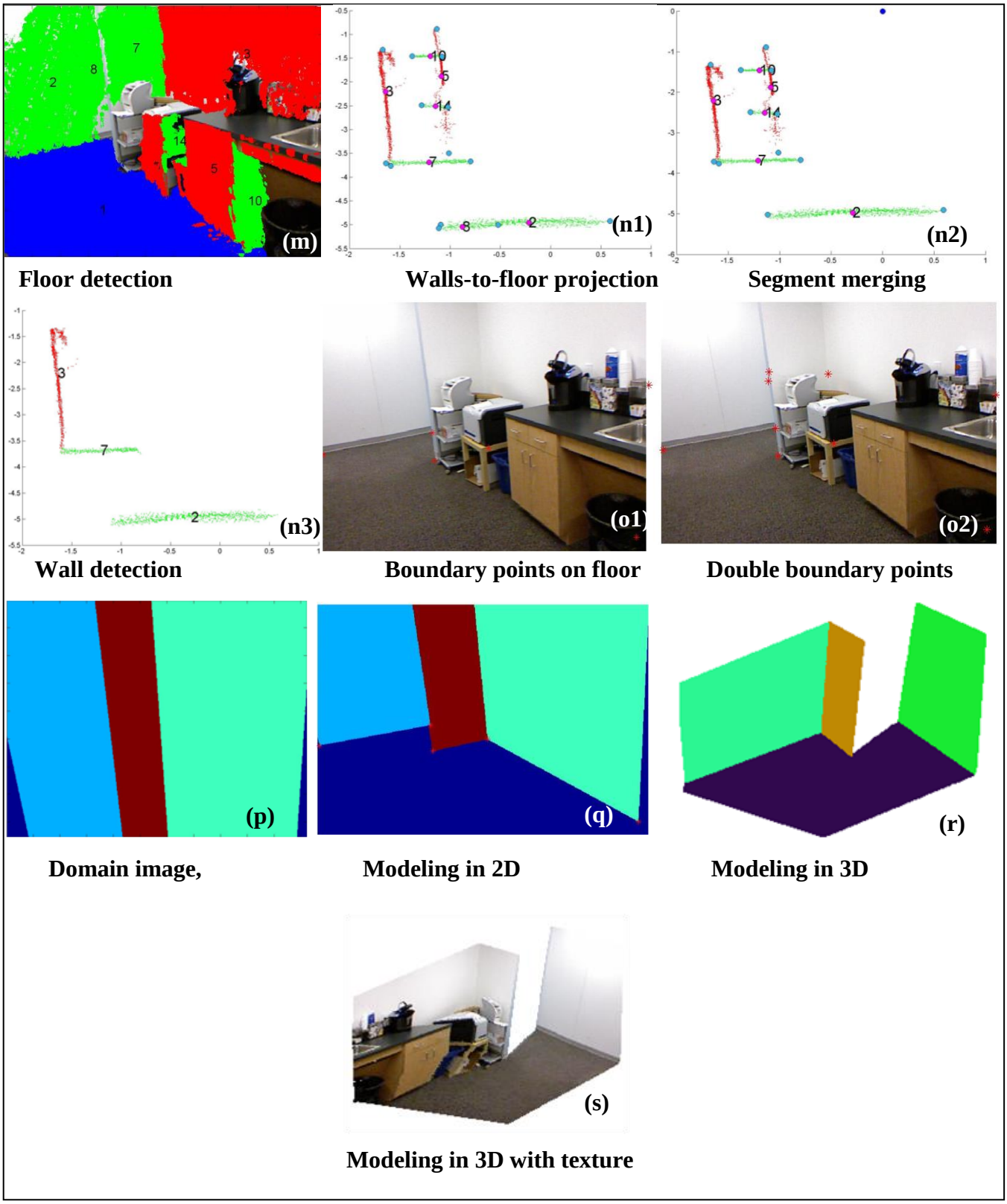


Figure 4. 11 The modeling pipeline – continued

Figures 4.12 and 4.13 show a synthetic data processing pipeline. Figure 4.12a and b are color 2D images and point clouds in 3D. Planar segmentation like RANSAC produces planar segmentation as shown in 4.12c. After alignment with major orientation, aligned planes appear as seen in 4.12d, from which the floor can be detected. Figure 4.12e shows the detected floor 10. When the reserved vertical planes are rotated so that we have 4.12f, the vertical planes are projected to thefloor plane. The vertical planes are shown as green and red segments, while the blue floor is parallel to screen. Red and green segments' visibility voting matrix can be calculated by comparing the visibility of segments to origin points. The visibility matrix determines whether a segment will be removed. The reserved segments are real walls, which is shown in 4.12g. Boundary points can be obtained in 4.12h by comparing adjacent segments. Figure 4.12i shows the boundary points projected back to 2D. Connecting boundary points will give boundary lines, which will produce the domain image as shown in Figure 4.12j. Each pixel in a domain will further be labeled whether is wall or floor depending on the wall and floor's plane depth comparisons, i.e., each pixel will have a ray intersecting the domain's wall plane and floor plane, which will produce two 3D points, e.g., (xd, yd, zd) being intersection points on the domain wall plane and (xf, yf, zf) intersection points on floor plane. Whether the pixel will be labeled as a domain wall or floor depends on depth zf and zd. Figure 4.12k gives the 2D segmentation after the above processing. Depending on the 2D segmentation (2D modeling), we can obtain 3D modeling (l).

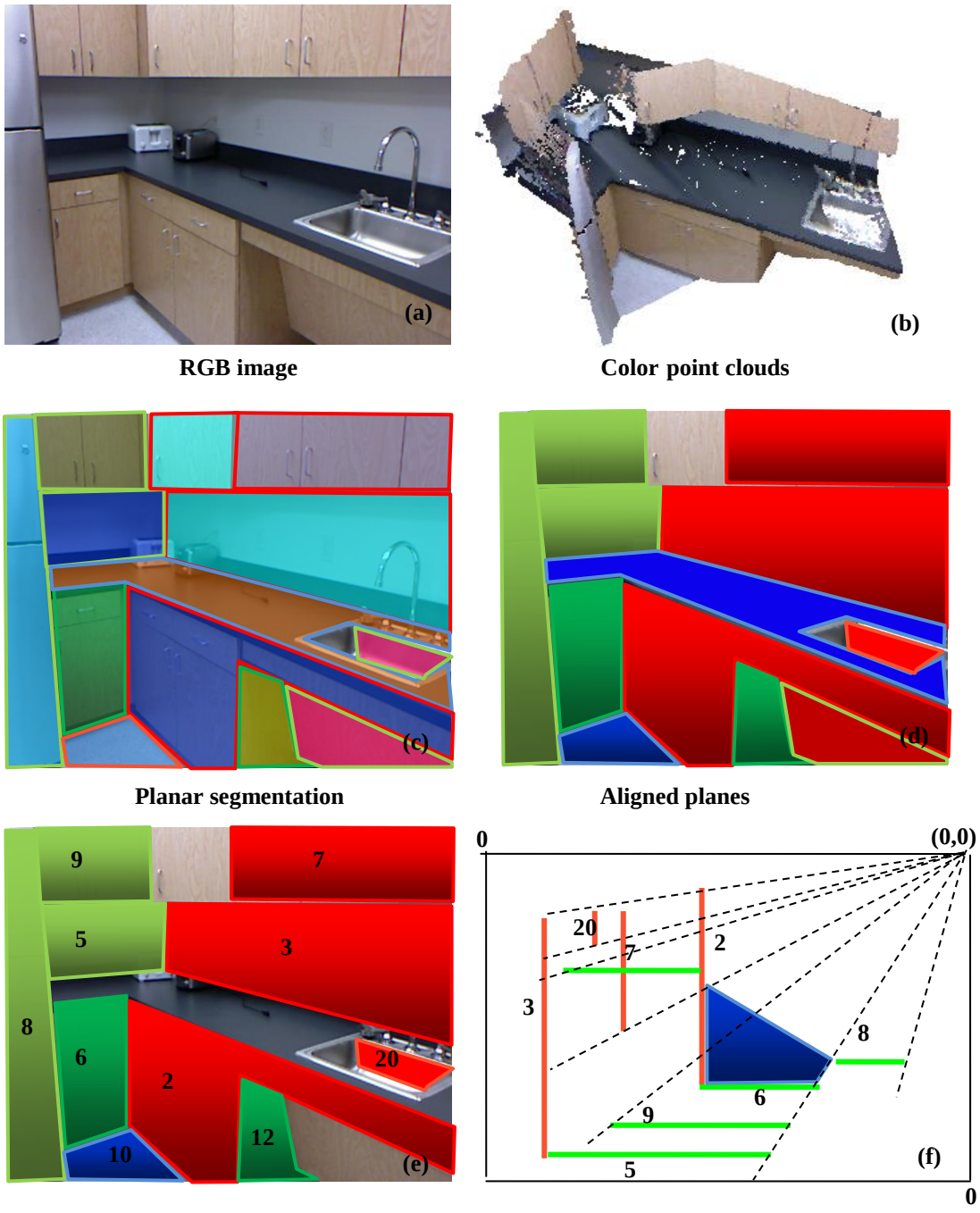


Figure 4. 12 Synthetic data processing pipeline

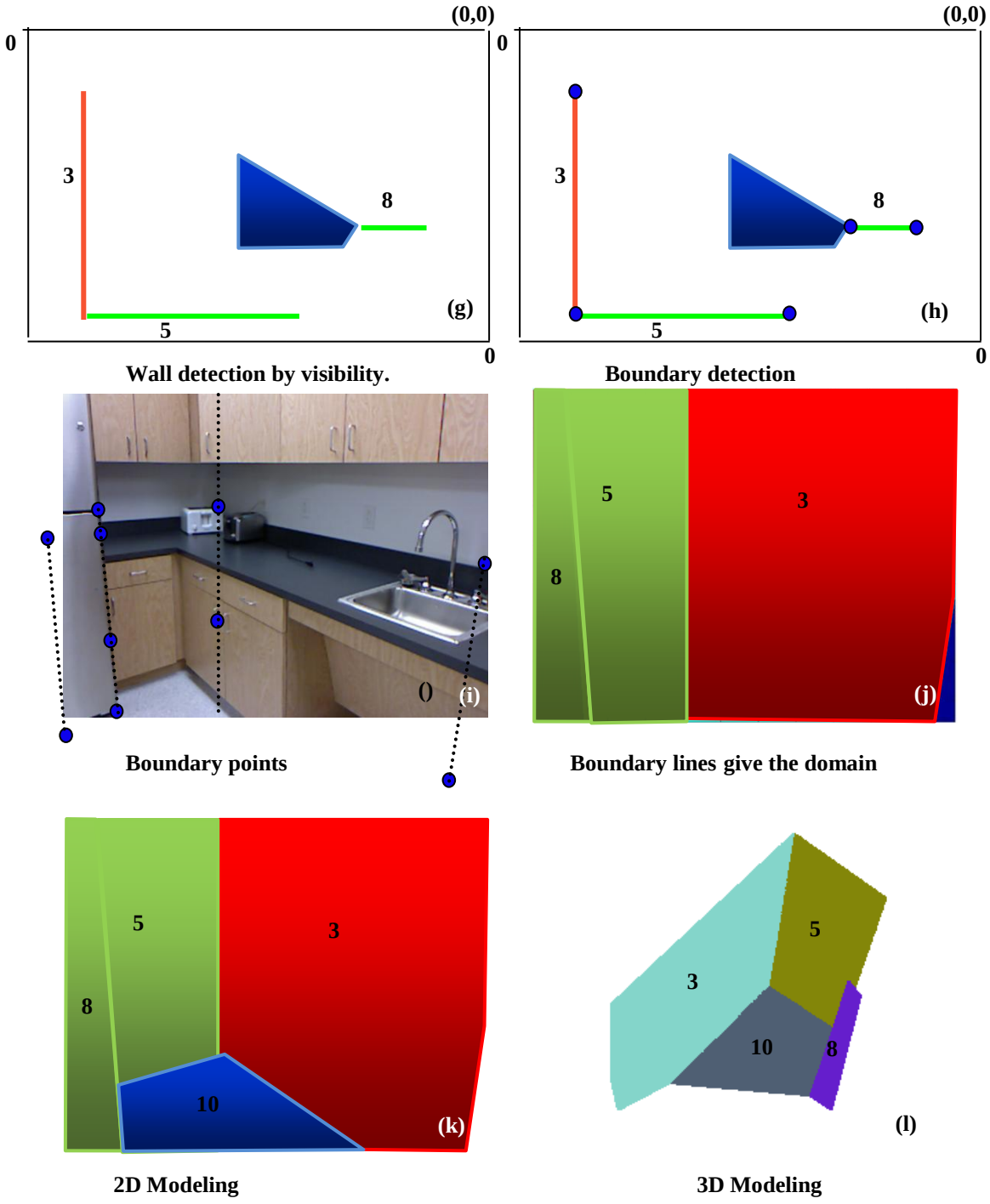


Figure 4. 13 The synthetic data processing pipeline-continued.

4.3 Modeling program

All of the previous processing were implemented in the MATLAB platform (MathWorks Inc.). Since the whole process is automatic, the user simply runs the program on selected Kinect data. The data sets used here are obtained from the NYU data webpage: http://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html. This section goes through the program.

The code follows the processing steps one by one. Users can select Kinect data for preprocessing, which shows the cropped RGB image and registered depth image as well as vanishing point detection results for RANSAC planar region detection, automatic floor, wall detection, automatic wall boundary point detection and domain image creation. Finally, 3D modeling is completed by extruding from the boundary points. The 3D modeling with color texture image is saved as an .obj file.

The code can be classified into several categories: 1) Basic RGBD processing including D to point clouds under the frame of an IR camera, point clouds from IR frame to RGB frame, point cloud projections onto RGB camera, hole filling, 3D normal calculation of point clouds, curvature calculation, vanishing point detections, and vanishing points to world axis transformation; 2) RANSAC planar region segmentation is another category discussed at length in Chapter 2; 3) Automatic floor, wall and wall boundary detection are other methods of modeling; 4) 3D Modeling by extruding from automatic detection of boundary points and floor planes. The code will automatically create a folder for each modeling case and save important intermediate processing results.

The resultant 3D model .obj file can be open with open source software MeshLab. It is downloadable from. http://meshlab.sourceforge.net/. The following figure shows the user

interface of MeshLab. With MeshLab, people can explore modeling and point clouds conveniently. An XYZ+RGB point cloud and modeling were mixed together. We can see that the modeling predicted the layout correctly. We can see the walls and ground beneath the tables microwave oven, and printer. There is a gap between the cyan wall and green wall, which could be a hallway.

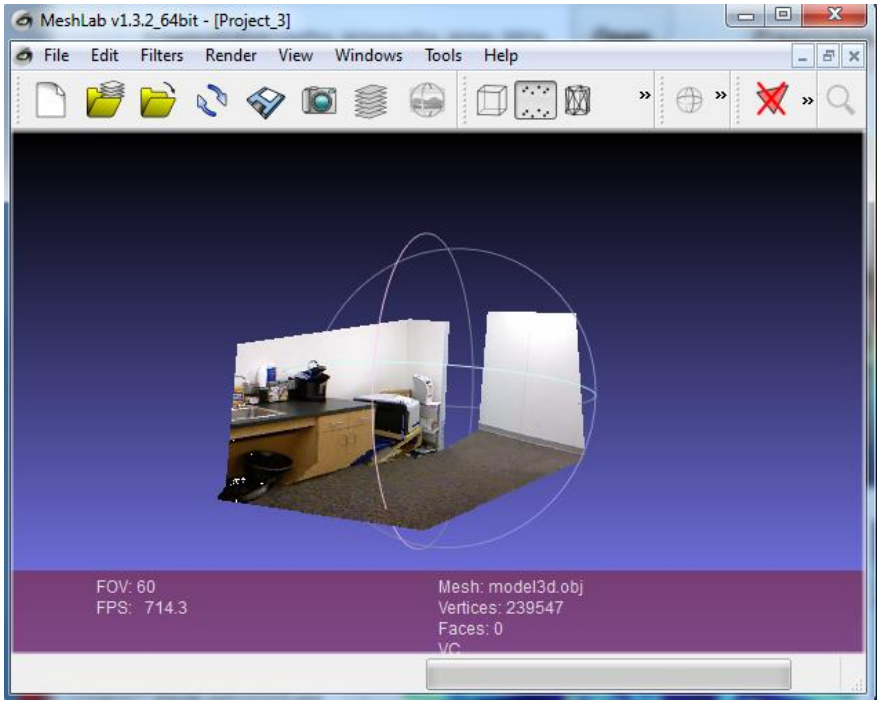


Figure 4. 14, 3D modeling open from MeshLab software. Here the color point clouds are mixed with the modeling.

4.4 3D Modeling results and discussion.

Seventy-two case testing results were done to test the performance of the algorithms. A total of 13 failed. The success rate was > 80%. Among the failures, five were caused by reconstruction failure. Another eight tests failed either at RANSAC planar region segmentation, i.e. no RANSAC results at all or fails at major orientation detection sites. Sometimes the obtained major orientation is of only two axes.

Results are shown in Figure 4.15. Reconstruction failure cases are shown in Figure 4.16. The testing indicates that the modeling method is robust and accurate. 4.15- a, d, g, j, m, p, s, v, y, and α are the RANSAC planar region segmentation results. Correspondingly, 4.15- b, e, h, k, n, q, t, w, z, and δ are the pure domain graph which shows the exact area of ground and walls. 4.15-c, f, i, l, o, r, u, x, and γ are the 3D modeling with color image texture.

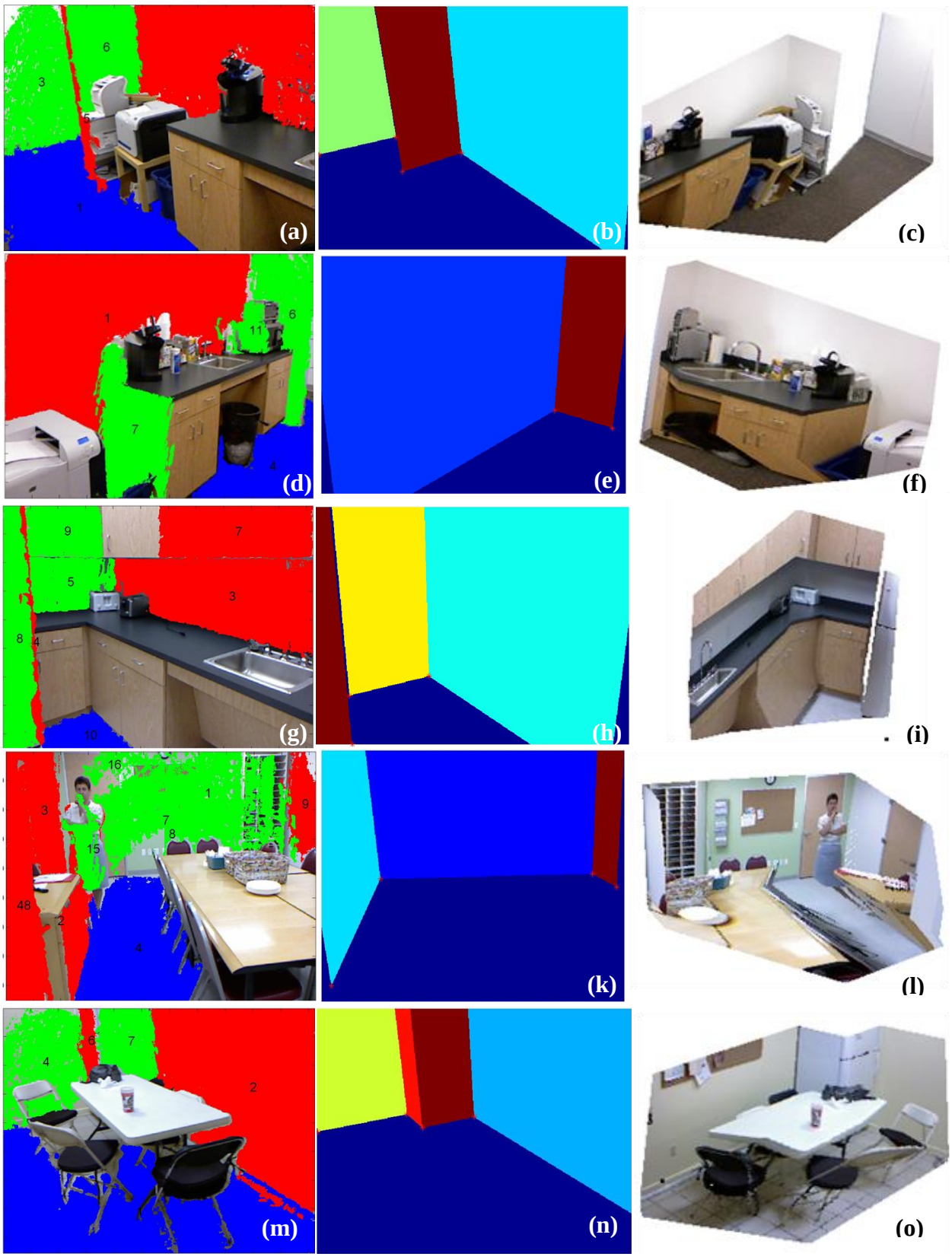


Figure 4. 15 Modeling of some scene images. Continuedon next page.

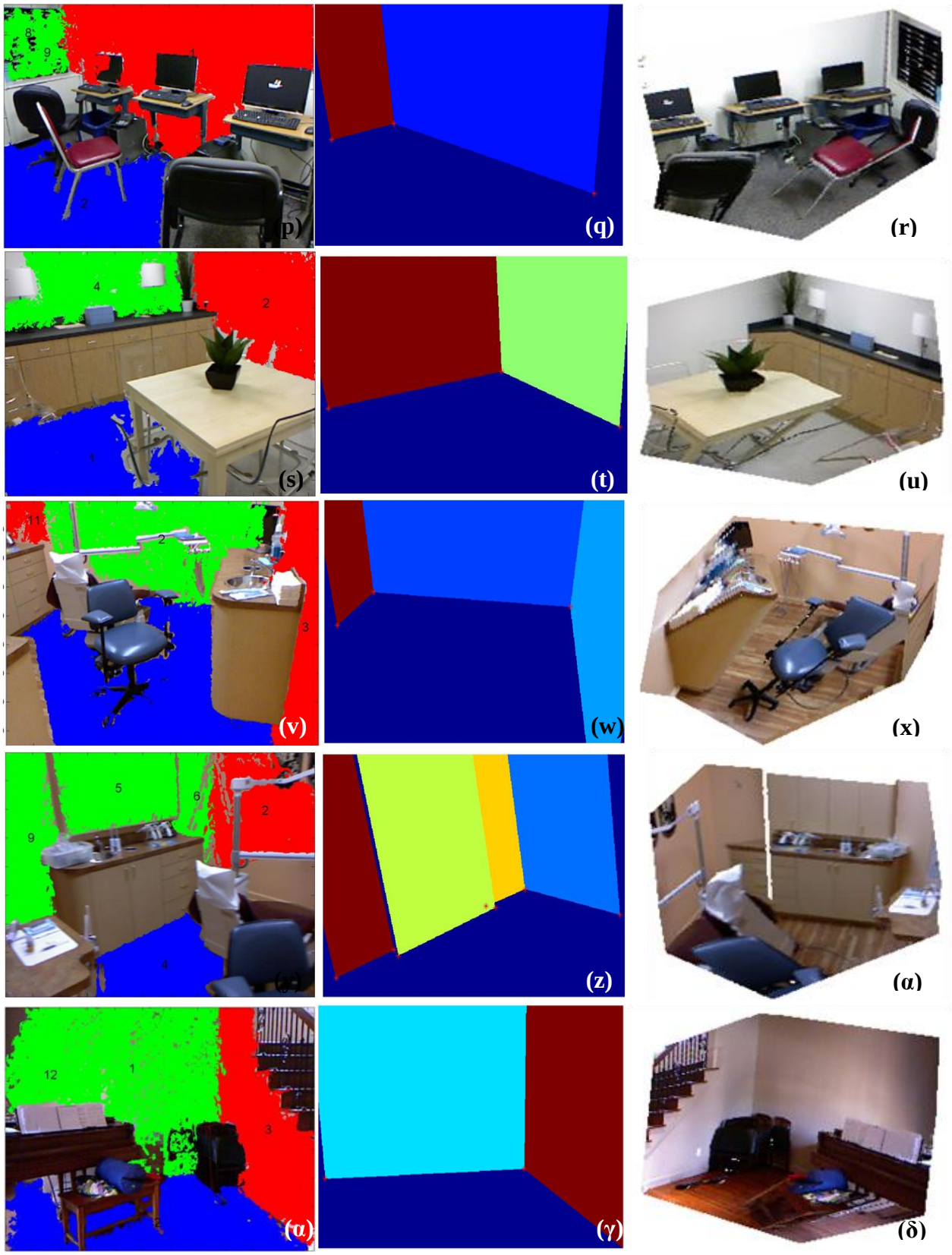


Figure 4.15 Continuation of the preceding test results.

The following Figures 4.16–4.20 gives a detailed explanation of failure cases and possible reasons of reconstruction failure. In the Figure 4.16a is the RANSAC results. Figure 4.16b is the planar region aligned with major orientation, where the first time cleaning was done. Walls are labeled as red and green, while the floor is labeled blue. Figure 4.16c projects walls to the floor. Figure 4.16d has automatic wall detection. Figure 416e has wall boundaries obtained from remaining walls of 4.16d that were projected to the floor to generate the first points and

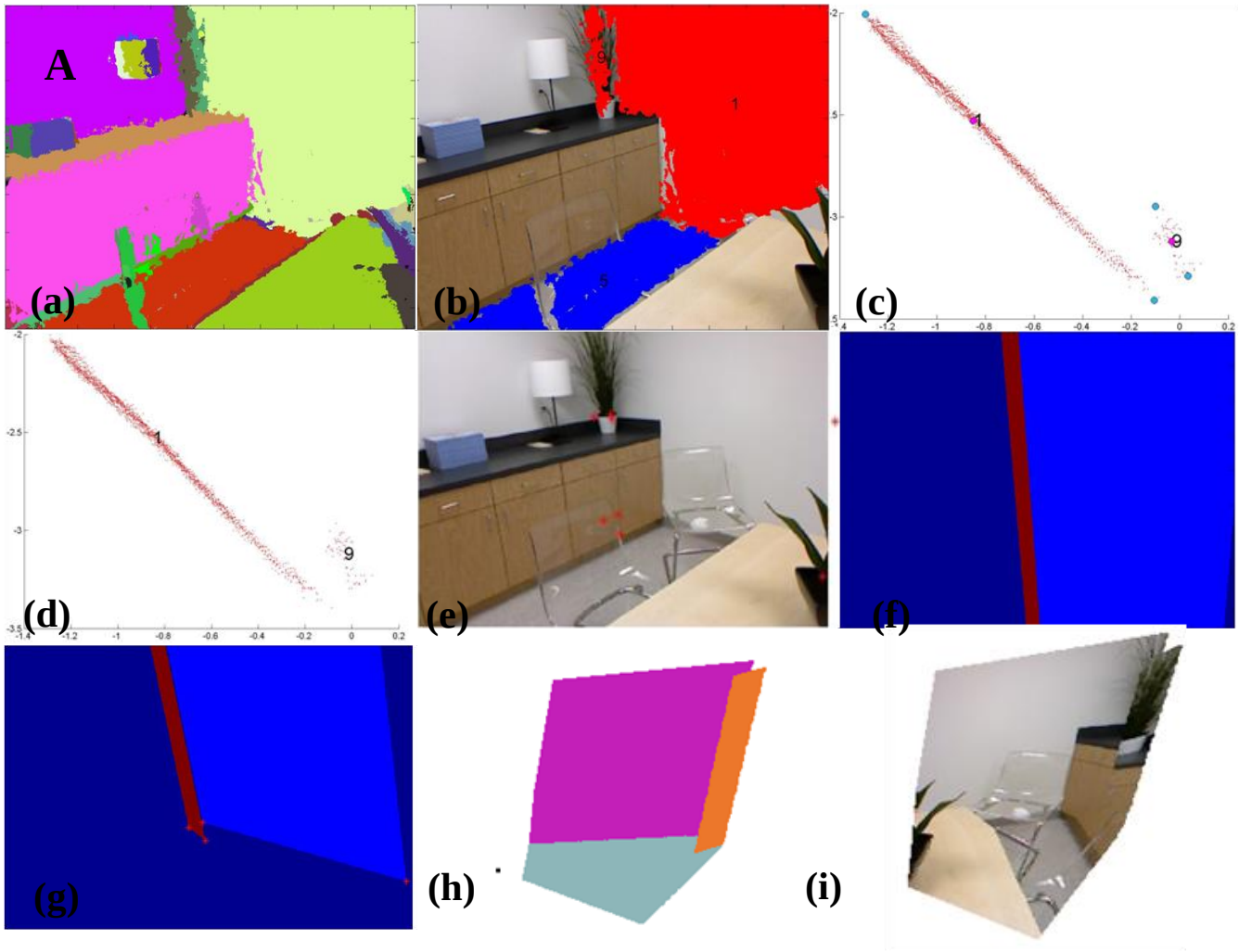


Figure 4. 16 Failed cases of modeling.

extruded to generate the second points along the wall boundary; Figure 4.16f shows the wall's domain image boundaries, and 4.16g shows 2D segmentation of wall and floor. Figure 4.16h shows 3D reconstruction, and 4.16I shows 3D reconstruction with texture.

Figure 4.16 failed at 4.16b where the normal vector of the planar region obtained from RANSAC will compare with three major orientation vectors. If the smallest angle between them is less than 30 degrees, then the planar region will be labeled to align with that orientation. Red label planes aligned with x; then, green label planes aligned with y; blue label planes aligned with vertical major orientation z. If the plane is not labeled to be aligned with any orientation, we can't increase the threshold to enclose the undetected regions because this will lead to more outliers, which will subsequently increase the difficulty of the following automatic wall region detection. So the main reason for this one is plane A is rejected at this step.

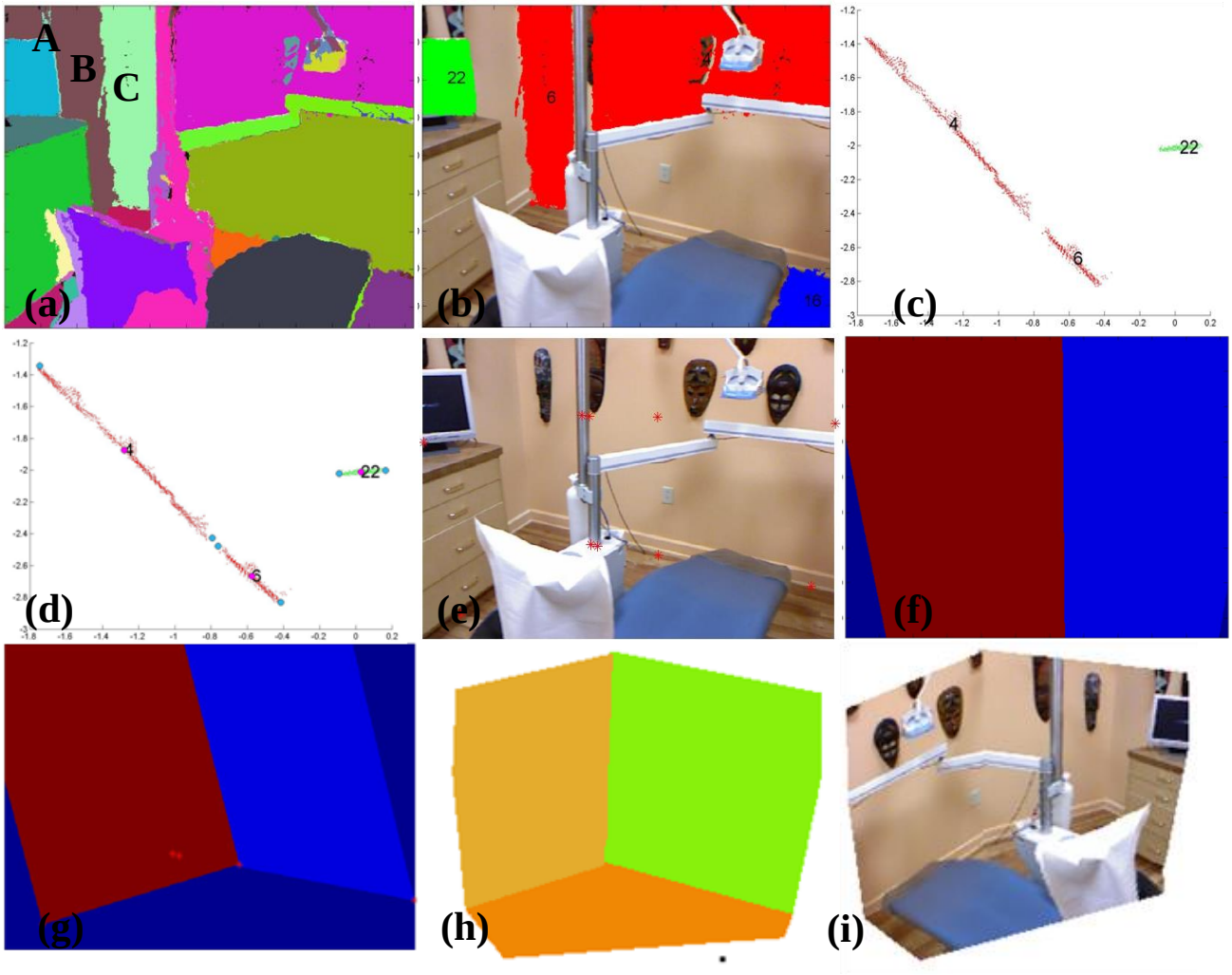


Figure 4. 17-failure cases continue.

Figure 4.17b failed for the same reason as Figure 4.16. So the main reason for this one is that planes A, B, and C were labeled in 4.17a as rejected at this step.

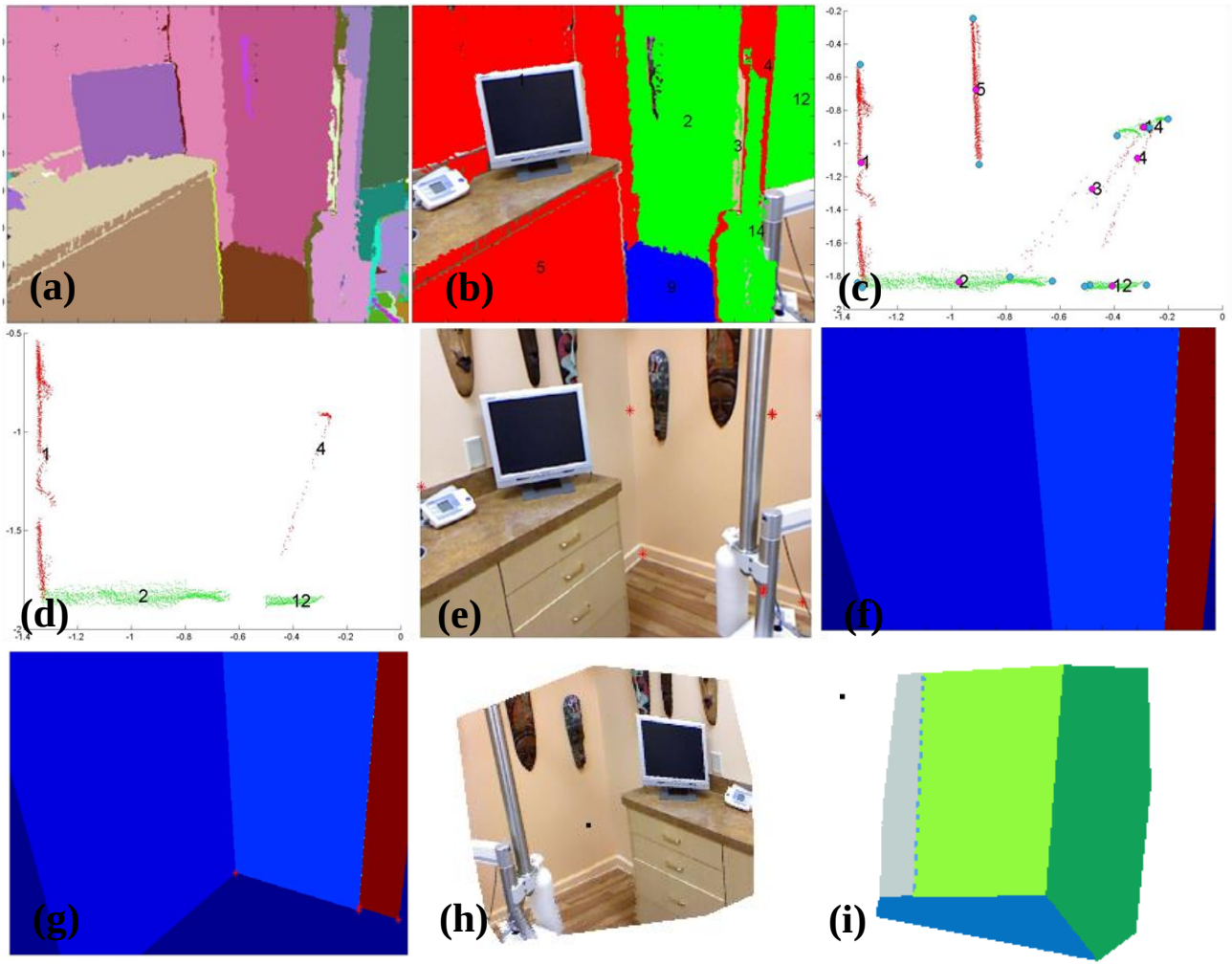


Figure 4. 18-Failure cases continue.

Figure 4.18d failed while Region 4 remained as a wall. In the automatic wall region detection, the visibility from the origin point (0,0) was measured. A matrix was maintained to record the vote of how many time a wall blocked other walls. If the blocking vote is non zero, then this wall should be removed. This will clean all the regions in front of the walls. Region 4 blocks no other walls, thus, it is reserved as a real wall.

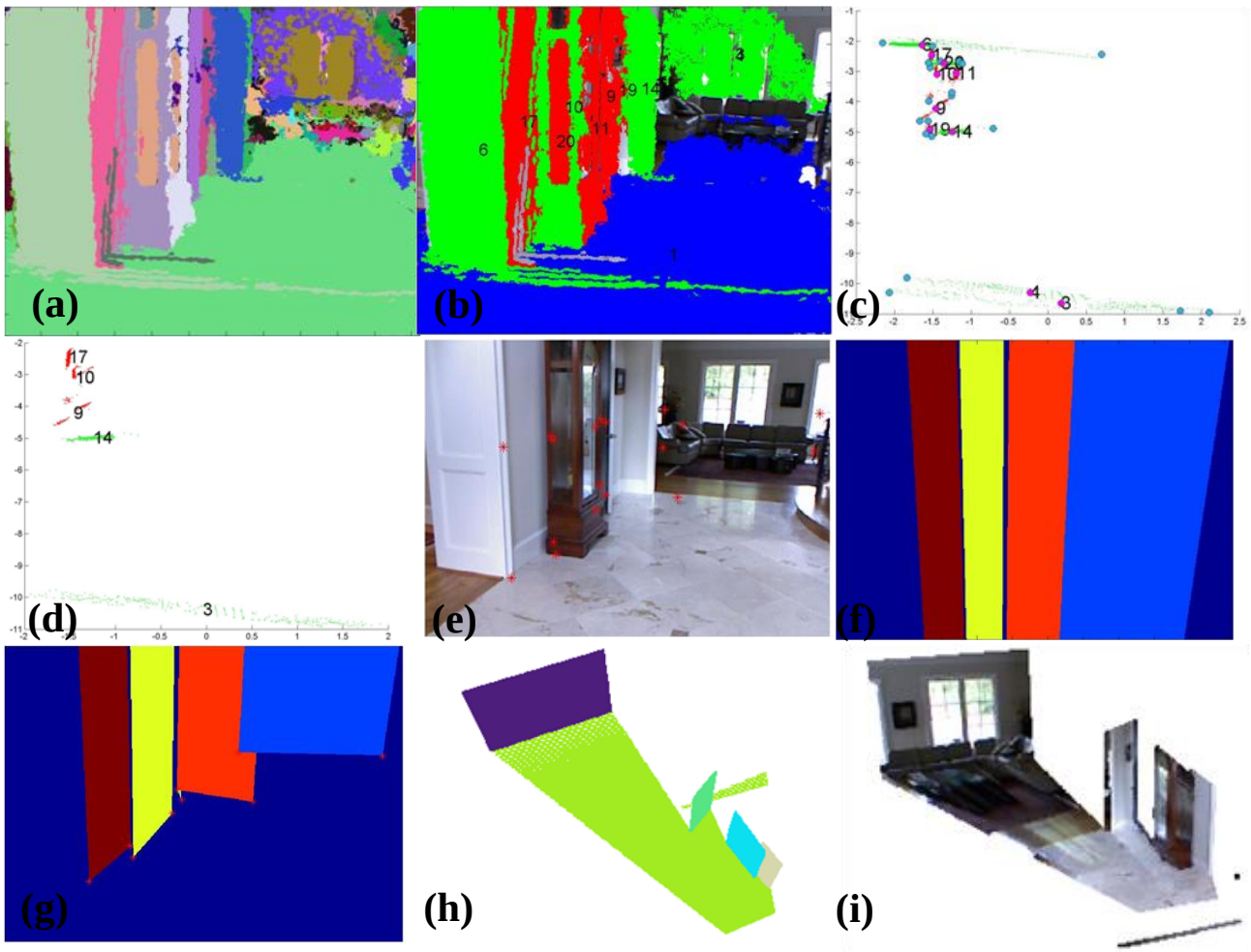


Figure 4. 19 Failed cases continue.

Figure 4.19 failed because of noisy point clouds. At large depth regions, the point clouds, of the planar regions are often noisy. The failure rate of detecting correct wall region become lower. This one belongs to this case. Noisy point clouds affect the RANSAC results, thereby directly affecting the automatic wall region detection.

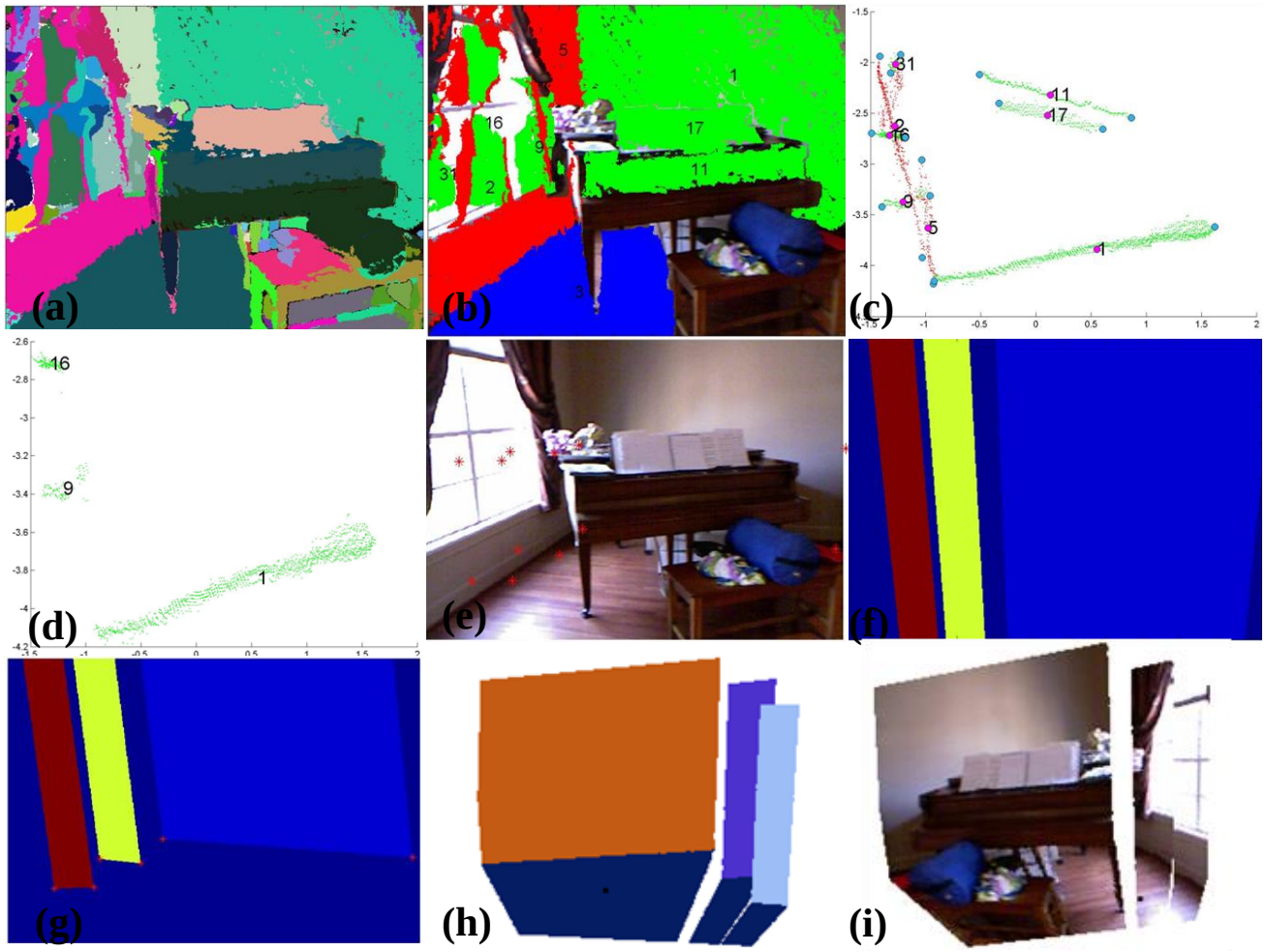


Figure 4. 20 Failed cases continue.

This one fails at 4.20d. From 4.20c, we can see that Regions 16 and 9 are blocked by Region 2. Region 2 is blocked by Regions 5 and 31; thus 16 and 9 remained, while 16 and 9 were from wrong regions such as window glass. Hence, a real wall region may be rejected because of unexpected outlier regions. The outlier in this figure may be from outside the window.

As can be seen from the processing, several failure reasons can be determined. First, noisy point clouds tend to give incorrect plane fitting. Far distances also cause poor 3D point clouds. Secondly, failure often occurs at automatic wall detection due to unexpected outlier regions..

CHAPTER 5. CONCLUSIONS

In this study, a modeling method of extruding from the automatically detected floor, wall, and wall boundary points was developed. Preprocessing the RGBD generated point clouds. Point clouds were then processed to have planar region segmentation with region growing method or RANSAC planar segmentation method. Compared with RANSAC, the region growing method gave relatively better results, but the speed was too slow. So the RANSAC method is preferred in context. Vanishing point detection on colors generally generates three vanishing points. These vanishing points can help to determine the world coordinate axes, i.e., the major orientation of the planes. This is equivalent to the major orientations obtained from planar segmentation. However, world axis from VPs has correct labeling with scenes content, which helps to get correct labeling of major orientation obtained from planar segmentation. Here the major orientation obtained from VPs was used, but it should be noted that there is no big difference between the two sets of major orientations. The major orientation determined the planar regions alignment and relations between them. The automatic floor and wall detection were done in 2D projection of potential wall planes to floor. Next, automatic boundary points were obtained, which was used to segment the wall domains graph. 3D modeling was performed with the domain graphs. The testing indicated that the modeling method was generic and very robust. However, there are still some cases where the modeling cannot work very well. Further improvements are necessary and more research will be aimed at making those improvements in the future.

REFERENCES

1. A. Criminisi, I. Reid, and A. Zisserman. “Single View Metrology.” Int. J. Comput. Vision 40, 2 (2000), 123-148.

2. A. Criminisi. Single-View Metrology: Algorithms and Applications. Proc., 24th DAGM Symposium on Pattern Recognition, Luc J. Van Gool (Ed.). Springer-Verlag, (2000) 224-239.

3. V. Hedau, D. Hoiem, and D. Forsyth. "Recovering the spatial layout of cluttered rooms." Proc., Computer vision, 2009 IEEE 12th international conference,. IEEE, (2009). 1849-1856.

4. DC. Lee, M. Hebert, and T. Kanade. "Geometric reasoning for single image structure recovery." In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, (2009), 2136-2143.

5. Coughlan, J.M., and A.L. Yuille. "Manhattan world: Compass direction from a single image by bayesian inference." In Computer Vision, 1999. Proc., Seventh IEEE International Conf., vol. 2, pp. 941-947. IEEE, 1999.

6. Coughlan, J.M., and A. L. Yuille. "The Manhattan world assumption: Regularities in scene statistics which enable Bayesian inference." NIPS, pp. 845-851. 2000.

7. Neverova, N., D. Muselet, and A. Trémeau. "21/2 D Scene Reconstruction of Indoor Scenes from Single RGB-D Images." Computational Color Imaging. Springer Berlin Heidelberg, 2013. 281-295.

8. N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. "Indoor segmentation and support inference from RGBD images." Computer Vision–ECCV 2012, pp. 746-760. Springer Berlin Heidelberg, 2012.

9. C.J. Taylor and A. Cowley. "Parsing indoor scenes using rgb-d imagery." Robotics: Science and Systems, vol. 8, pp. 401-408. 2013.

10. M.A. Fischler, and R.C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." Communications of the ACM 24, no. 6 (1981): 381-395.

11. J. Lezama, R.G. von Gioi, G. Randall, and J.M. Morel. "Finding vanishing points via point alignments in image primal and dual domains." Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference, pp. 509-515. IEEE, 2014.

12. Y. Ma, ed. An invitation to 3-d vision: from images to geometric models. Vol. 26. Springer Science & Business Media, 2004.

13. R. Hartley, and A. Zisserman. Multiple view geometry in computer vision. Cambridge university press, 2003.

14. D. C. Brown. "Decentering distortion of lenses." Photogrammetric engineering. 32 (3) (2006) 444–462.

15. Conrady, A. E. "Decentered." Monthly notices of the Royal Astronomical Society 79 (1919): 384–390.

16. E. Guillou, D. Meneveaux, E. Maisel, and K. Bouatouch. "Using vanishing points for camera calibration and coarse 3D reconstruction from a single image." The Visual Computer, 16:396-410, 2000.

17. R. Cipolla, T. Drummond, and D. Robertson. Camera calibration from vanishing points in images of architectural scenes. BMVC99, pages 382-391, 1999.

18. B. Caprile, V. Torre, Using vanishing points for camera calibration. Inter. J. Computer vision, 4(2), 127-139, 1990.

19. B. Freedman, A. Shpunt, and Y. Arieli. "Distance-varying illumination and imaging techniques for depth mapping." U.S. Patent No. 8,761,495. 24 Jun. 2014.

20. Zalevsky, Zeev, et al. "Method and system for object reconstruction." U.S. Patent No. 8,400,494. 19 Mar. 2013.

21. A. Shpunt, Z. Zalevsky, "Three-dimensional using speckle patterns" US 12/282,517, Mar 8, 2007

22. R. Grompone von Gioi, J. Jakubowicz, J. M. Morel, and G. Randall. LSD: A line segment detector. Image Processing On Line, 2012.

23. Matessi, A., and L. Lombardi. "Vanishing point detection in the Hough transform space." In Euro-Par'99 Parallel Processing, pp. 987-994. Springer Berlin Heidelberg, 1999.

24. M. Dubska, A. Herout, and J. Havel, PClines - Line Detection Using Parallel Coordinates, Proc., CVPR 2011, IEEE Computer Society, 2011,1489—1494.

25. M. d'Ocagne, Coordonnées parallèles et axiales : Méthode de transformation géométrique et procédé nouveau de calcul graphique déduits de la considération des coordonnées parallèles. Paris: Gauthier-Villars, 1885.

26. J.J. Ding, The class of "Advanced Digital Signal Processing," Department of Electrical Engineering, National Taiwan University (NTU), Taipei, Taiwan, 2008.

27. W. K. Pratt, Digital Image Processing 4th Edition, John Wiley & Sons, Inc., Los Altos, California, 2007

28. T. Rabbani, H. Frank, and G. Vosselmann. "Segmentation of point clouds using smoothness constraint." International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 36.5 (2006): 248-253.

29. J. Strom, A. Richardson, and E. Olson. "Graph-based segmentation for colored 3D laser point clouds." Proc., Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, pp. 2131-2136. IEEE, 2010.

30. L.E. Peterson, "K-nearest neighbor." Scholarpedia 4.2 (2009): 1883.

31. J.L. Bentley, "Multidimensional binary search trees used for associative searching." Communications of the ACM 18.9 (1975): 509-517.

32. T.T. Htar, and S.L. Aung. "Enhancement of Region Merging Algorithm for Image Segmentation." Proc., International Conference on Advances in Engineering and Technology (ICAET. 2014.)

33. A. Tinku and K.R. Ajoy, Image Processing: Principles and Applications. ISBN: 978-0-471-71998-4

34. R.B. Fisher, and D. K. Naidu. "A comparison of algorithms for subpixel peak detection." Image Technology. Springer Berlin Heidelberg, 1996. 385-404

35. R. Schnbel, R Wahl and R, Klein, "Efficient RANSAC for point cloud shape detection." Proc., Computer Graphics Forum, 2.26 (2007): 214-226.