# Clutter Detection in Indoor Scenes Using Structure 3-D Sensor

*Zhiji Liu*


Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

May 15, 2015

Technical Report No. ECE-2015-04

# Summary

Hoarding is a psychological disorder that manifests itself as a compulsive need to collect and keep unnecessary and useless items in living quarters. It may have a very negative impact on people's quality of life. Bacteria, insects and even rodents can live in such an environment and spread disease. Furthermore, a room full of useless items can prevent people from moving freely and leading normal life. Healthcare professionals and researchers who study hoarding usually visit a patient's room and take pictures to compare them with an image database in order to judge the degree of hoarding []. However, such a comparison is subjective and often unreliable.

The goal of this MS project is to develop computational algorithms to automatically assess the degree of hoarding. We believe this is the first effort to develop such an automatic, objective, real-time hoarding assessment tool ever. One possible approach is to compute the percentage of the image that clutter occupies and decide the hoarding severity based on this percentage. Thus, the main focus of this particular project is the detection of clutter in indoor images. The fundamental assumption we make is that real-life clutter corresponds to high density of edges in a captured image.

First, we develop a 2-D method that computes average magnitude of luminance gradient over small image blocks. Thresholding this magnitude leads to the detection of clutter (average magnitude below a thresholds is deemed as non-clutter). Since this method results in many false positives especially in flat but textured areas (e.g., busy wallpaper), we also use a 3-D sensor to capture depth of the room. The second method we develop uses local planarity estimation over small blocks (thresholding of the variance of the magnitude of derivatives). Although combined with the 2-D method it improves the final decisions, the method is at times unreliable due to the use of little data. Finally, we develop a global planarity estimation method based on plane fitting using RANSAC algorithm. Any depth areas that do not fit planar structure are deemed as outliers or clutter. By fusing the results of the 2-D method with those of the global plane fitting, we further improve the results although the method occasionally results in catastrophic failures due to random initialization.

.

# Table of Contents

# 1 Introduction

Hoarding, which is known as the behavior of keeping unnecessary and useless items in living quarters, has a negative effect on people's quality of life [1-3]. Bacteria, insects and even rodents can live in such an environment and spread disease. Furthermore, a room full of useless items can prevent people from moving freely and leading normal life. Researchers who study hoarding usually visit a patient's room and take pictures. Then, they compare these images with an image database to judge the degree of hoarding [4-7]. However, such a comparison is subjective and often unreliable. The goal of this MS project is to implement an algorithm that can immediately and automatically assess the degree of hoarding, for example by computing the percentage of the image that clutter occupies. Thus, the main focus of this work is the detection of clutter in indoor images.

We note that hoarded items in a room usually correspond to high-frequency areas in the captured image (e.g., lots of edges at random orientations – see Figure 1). This suggests that some form of gradient thresholding may be useful in detecting clutter areas. However, some areas void of clutter, like wallpaper-covered walls, paintings, posters may also exhibit fine detail. Thus, using only gradient computation could result in excessive false alarms. Clearly, the capture of 2-D brightness and color only is not sufficient in this case. Therefore, we propose to explore using a 3-D sensor that, in addition to brightness and color, also captures depth information. The knowledge of depth (structure) is expected to prove useful in disambiguating textured, but flat, patterns (wallpaper) from clutter that exhibits non-flat structure. Therefore, our final goal is to combine the traditional 2-D technique with 3-D information to accurately detect areas of clutter.

# 2 Assumptions and Project Statement

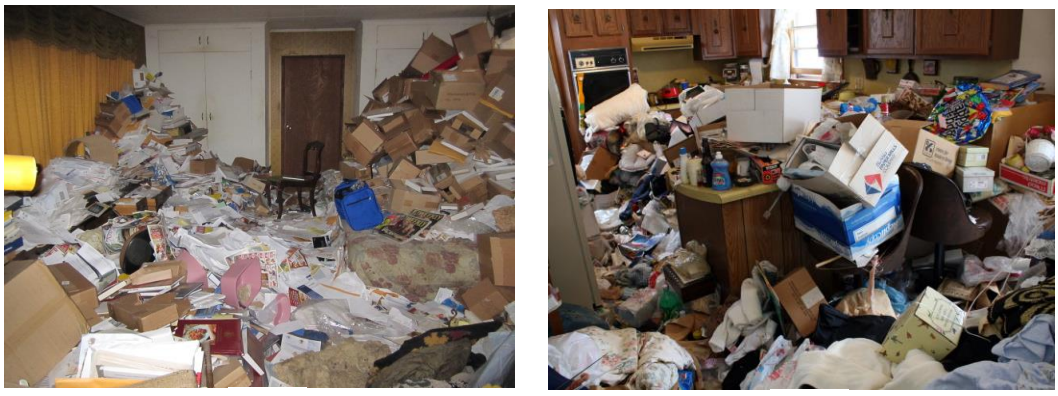
As in any image processing algorithm, we make several assumptions on the input images in order to make the problem more constrained and thus feasible. The assumptions we make regarding images are as follows:

1. The image of a room should contain at least two walls: two side walls, or one side wall and either floor or ceiling (see figure below).

(a) (b)

*Figure 1 Two examples of a hoarder's room: (a) image satisfying assumption 1 (two side walls and ceiling are visible); (b) image violating this assumption.*

2. In order to estimate wall/ceiling/floor orientation, a significant area of the wall/ceiling/floor must be visible (void of clutter).



(a) (b)

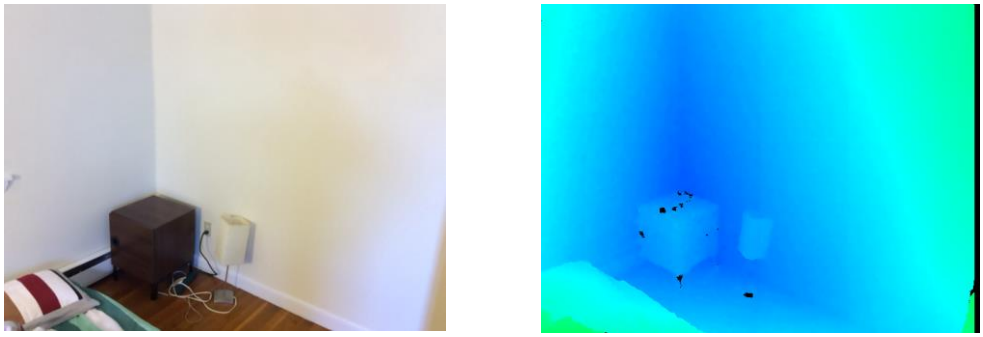
*Figure 2 Two examples of a hoarder's room: (a) image with sufficiently visible two side walls; (b) image with insufficient wall area visible.*

3. The camera should be aligned horizontally with the room. This simplifies wall orientation estimation and makes the algorithm more robust.

(a) (b)

*Figure 3 Example of: (a) proper alignment of a camera with room floor, and (b) improper alignment.*

4. The camera should be pointed at the farthest corner of the room so that the intersection area of, for example, two side walls is more or less central in the field of view of the camera. This increases the robustness of wall orientation estimation.



(a) (b)

*Figure 4 (a) Image of a room with a corner between two walls being the furthest points of the scene from the camera, and (b) the corresponding depth map (dark blue indicates large depth).*

We characterize the indoor objects into three classes based on their surface texture and surface shape:

Type I: Low-texture objects with planar surface, e.g., walls, boxes (considered to be clutter), furniture, etc.)

Type II: High-texture objects with planar surface such as wallpapers, paintings, etc.

Type III: High-texture objects with non-planar surface, i.e., clutter.

The goal is to estimate what proportion of a captured image (field of view of the camera) does the clutter occupy, where wallpaper, paintings, bookshelves are not considered clutter.

# 3 Thresholding the Magnitude of Luminance Gradient

We assume that all input images have been scaled to the dimension of 640 pixels horizontally and 480 pixels vertically. We start by dividing the whole image into 20 by 20 non overlapping blocks, with an *n*-th block denoted by $\Lambda_n$. Inside each block $\Lambda_n$, we compute the horizontal derivative as follows: $d_n^H(i,j) = \Lambda_n(i,j) - \Lambda_n(i+1,j)$ and similarly the vertical derivative as: $d_n^V(i,j) = \Lambda_n(i,j) - \Lambda_n(i,j+1)$. Then, we compute gradient magnitude for each pixel in block $\Lambda_n$: $d_n(i,j) = \sqrt{(d_n^H(i,j))^2 + (d_n^V(i,j))^2}$. The overall magnitude of the gradient for the entire block is computed by summing up all the magnitudes of the gradient for each pixel:

$$d_n = \sum_{(i,j)\epsilon\,\Lambda_n} d_n(i,j)$$

In our case (640 by 480 image) we finally obtain a 32 by 24 matrix of gradient magnitudes that we threshold as follows:

$$\Gamma_1 \begin{matrix} \varepsilon \\ < \\ > \\ \eta \end{matrix} d_n$$

to make a clutter versus non-clutter decision for each block, where $\varepsilon$ denotes a potential clutter block and $\eta$ denotes a potential non-clutter block.

## 3.1 Implementation

Below are detailed steps of our implementation:

1. Resize image *I* to 640 by 480 pixels, and convert to grayscale
2. Divide the image into 20 by 20 blocks, with block number *n* denoted $\Lambda_n$
3. Inside each block, compute the horizontal and vertical derivative matrix: $d_n^H(i,j)$ and $d_n^V(i,j)$ as explained above.
4. Compute gradient magnitude in each block: $d_n(i,j) = \sqrt{(d_n^H(i,j))^2 + (d_n^V(i,j))^2}$
5. Sum up magnitude of the gradient in each block:

$$d_n = \sum_{(i,j)\epsilon\,\Lambda_n} d_n(i,j)$$

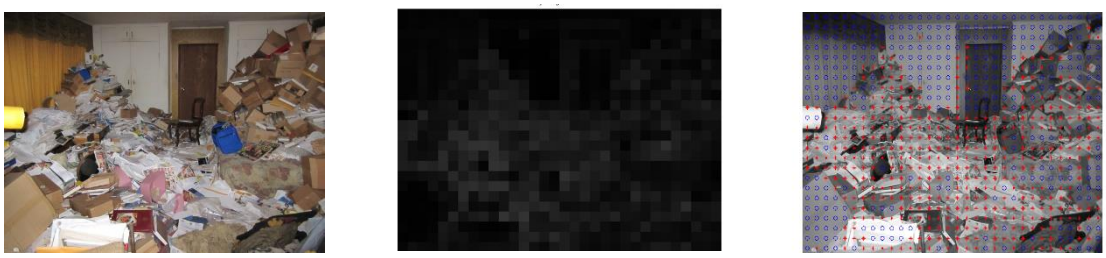
6. Set up a threshold value for the gradient magnitude, and make the binary decision:

$$\Gamma_1 \begin{matrix} 1 \\ < \\ > \\ 0 \end{matrix} d_n$$

7. The binary decision results in a binary labeling matrix $\Im_1$ containing elements of either 1 (clutter) or 0 (non-clutter)

## 3.2 Results

Figure 5 below shows our algorithm's output. Note numerous cardboard boxes with uniformly colored walls in the pile. This results in missed detections. At the same time, the door frame shows clear edges between the white wall and wood on the frame. These edges are detected and the corresponding blocks are falsely classified as clutter.



(a) (b) (c)

*Figure 5 (a) Input image (640 by 480); (b) Total gradient magnitude for each bock, the brighter the block the larger the total gradient magnitude $d_n$ in the block: (c) Final estimate of cluttered areas: red crosses denote clutter, blue circles denote no clutter.*

Since surface texture determines the final clutter versus non-clutter decision, indoor flat objects with texture such as paintings will affect the detection accuracy. Figure 6 shows an example of very inaccurate detection. Since the image contains a fine-texture carpet, it is detected as clutter.


(a) (b) (c)

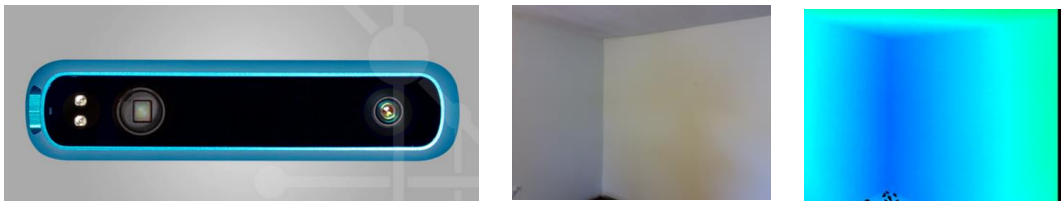
*Figure 6 (a) Input image (640 by 480); (b) Total gradient magnitude for each bock, the brighter the block the larger the total gradient magnitude $d_n$ in the block: (c) Final estimate of cluttered areas: red crosses denote clutter, blue circles denote no clutter.*

We can see that in Figure 6.c the high-texture carpet is considered to be clutter by our method. On the other hand, we can see in the bottom-left part of Figure 5.c that the boxes are detected as non-clutter due to their uniform coloring. As a result, in order to overcome the effect of surface texture, we need to use additional information, such as depth of the scene.

# 4 Thresholding the Variance of Depth

## 4.1 Structure Sensor

The Structure Sensor [8] is a tool that allows to capture dense geometry of objects using an infrared beam. The sensor provides distance information to objects in a scene, which in our case cab useful for distinguishing clutter from non-clutter. Figure 7 below shows the physical shape of the Structure Sensor that needs to be attached to iPad Air and connected via USB, and an output depth map.



(a) (b) (c)

*Figure 7 (a) Structure Sensor that needs to be attached to iPad Air and connected via USB; (b) depth as an RGB image captured by the sensor; (c) converted depth map.*

However, the information provided by the Structure Sensor is indirect. It provides the distance information in an RGB image format. We have to convert the RGB image into a distance matrix (depth map). In fact, the Structure sensor first detects the distance and then converts the distance into an RGB image as the output in order to provide a vivid (but not very useful) display to users. Here is the main procedure of conversion from distance to RGB depth image used by Structure Sensor:

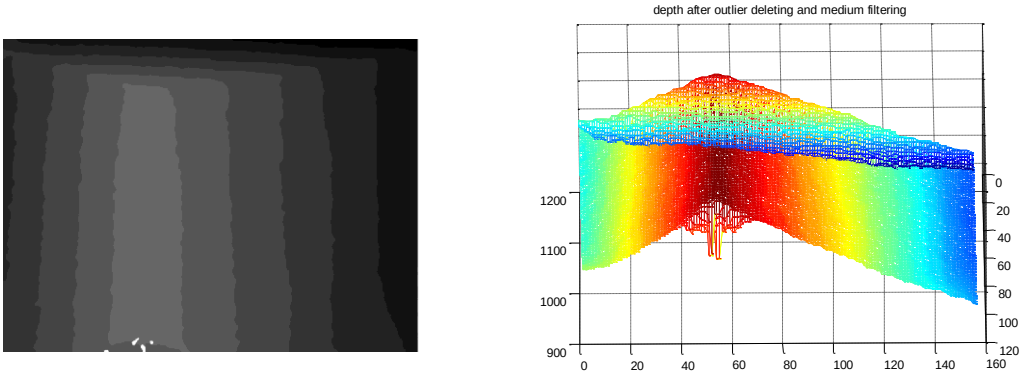
1. Measure the distance between room objects and the camera based on travel time of an infrared beam; the measured distance is non-linearly related to the true distance.
2. Use a look-up table to make the non-linear input values (distance) vary more linearly with true depth.
3. Convert the almost-linear distance into a 16-bit pattern; depict the first 8 bits as an upper byte, and the last 8 bits as a lower byte.
4. Choose the base colors as follows: White (closest), Red, Orange, Yellow, Green, Cyan, Blue, Black (farthest) based on the upper byte.
5. Use the lower byte to scale between the base colors.

Below is part of the original source code of Structure Sensor written in C++ to illustrate the above discussion. The code uses the upper byte and lower byte to convert distance information into RGB color. The output coloredDepthBuffer stands for the corresponding RGB color of a pixel.

```
" switch (upperByte)
    {
        case 0:
            _coloredDepthBuffer[4*i+0] = 255;
            _coloredDepthBuffer[4*i+1] = 255-lowerByte;
            _coloredDepthBuffer[4*i+2] = 255-lowerByte;
            _coloredDepthBuffer[4*i+3] = 255;
            break;
        case 1:
            _coloredDepthBuffer[4*i+0] = 255;
            _coloredDepthBuffer[4*i+1] = lowerByte;
            _coloredDepthBuffer[4*i+2] = 0;
            break;
        case 2:
            _coloredDepthBuffer[4*i+0] = 255-lowerByte;
            _coloredDepthBuffer[4*i+1] = 255;
            _coloredDepthBuffer[4*i+2] = 0;
            break;
        case 3:
            _coloredDepthBuffer[4*i+0] = 0;
            _coloredDepthBuffer[4*i+1] = 255;
            _coloredDepthBuffer[4*i+2] = lowerByte;
            break;
        case 4:
            _coloredDepthBuffer[4*i+0] = 0;
            _coloredDepthBuffer[4*i+1] = 255-lowerByte;
            _coloredDepthBuffer[4*i+2] = 255;
            break;
        case 5:
            _coloredDepthBuffer[4*i+0] = 0;
            _coloredDepthBuffer[4*i+1] = 0;
```

```
            _coloredDepthBuffer[4*i+2] = 255-lowerByte;
            break;
        default:
            _coloredDepthBuffer[4*i+0] = 0;
            _coloredDepthBuffer[4*i+1] = 0;
            _coloredDepthBuffer[4*i+2] = 0;
            break;
    }"
```

In order to obtain distance (depth) from the RGB images provided by Structure Sensor, we need to reverse the above operation. Figure 8 below shows the result of this reverse operation.




(a) (b)

*Figure 8 Depth (distance) map of a room obtained from RGB image that encodes depth produced by Structure Sensor: (a) as grayscale (the brighter the pixel, the larger the distance/depth); (b) the same depth shown as a surface plot from MATLAB.*

However, there are some drawbacks to the use of the Structure Sensor at the current stage: the depth information provided is not fully complete. We can see from the lower part of Figure 7 that there are some invalid black dots in the RGB image encoding depth, which is caused by the absorption of infrared light by the environment. Also, the depth map is not exactly linear compared to the true distance. This non-linearity might cause inaccuracies in plane orientation estimation which will be introduced soon.

## 4.2 Approach

Usually, the depth surface in a cluttered area has an irregular shape. On the other hand, this surface in a non-cluttered area is expected to exhibit regular shape (e.g., planar). For example, the depth matrix of a wall area is either steadily increasing in horizontal direction or in vertical direction. We can use this characteristic to try to eliminate some of the false alarms caused by the previous method of thresholding the magnitude of luminance gradient.

We first extract depth information from the Structure Sensor by converting the 640 by 480 by RGB image from the sensor into a 2-D 640 by 480 depth matrix *D*. This matrix contains the relative depth of the scene. We divide the depth matrix into 20 by 20 blocks $\Lambda_n$ as before. Inside each block, we compute horizontal and vertical derivative of the block element and take the absolute value as follows:

$$d_n^H(i,j) = \left|\Lambda_n(i,j) - \Lambda_n(i+1,j)\right| \quad \text{and} \quad d_n^V(i,j) = \left|\Lambda_n(i,j) - \Lambda_n(i,j+1)\right|.$$

We reshape the horizontal and vertical matrices of these derivatives into vectors of size of 1 by 400:

$$d_n^H(i,j) \xrightarrow{reshape} d_n^H(k) \quad \text{and} \quad d_n^V(i,j) \xrightarrow{reshape} d_n^V(k)$$

For each vector, we compute the variance as follows:

$$\text{var}_n^H = \frac{\sum_k (d_n^H(k) - \mu_{d_n^H})^2}{400} \quad \text{and} \quad \text{var}_n^V = \frac{\sum_k (d_n^V(k) - \mu_{d_n^V})^2}{400}$$

where $\mu$ denotes the corresponding mean.

As a result, we obtain a 32 by 24 matrix of variances of magnitudes of depth derivatives.

We then set up a threshold value $\Gamma_2$ for these variance matrices. If the variance of a block is lower than the threshold, then the corresponding frame is considered to depict a smooth surface area, which is either object of type I or type II (see Section 2). Similarly as in the 2-D method, we make the binary decision as follows:

$$\Gamma_2 \begin{matrix} \varepsilon \\ < \\ > \\ \eta \end{matrix} \text{var}_n^{\text{H}} \text{ or } \text{var}_n^{\text{V}}$$

and finally obtain a label matrix $\Im_2$. Since we are using this method to eliminate potential false alarms, we should keep all non-clutter areas detected by the 2-D method unchanged. Therefore, we combine the results of the two methods by simply performing a dot-matrix dot multiplication of $\Im_1$ and $\Im_2$ (AND operator).

## 4.3 Implementation

Below are detailed steps of our implementation:

1. Convert the RGB depth image into depth map *D*
2. Divide depth map into 20 by 20 blocks $\Lambda_n$
3. In each block compute the horizontal and vertical derivatives and take the absolute value: $d_n^H(i,j) = \left|\Lambda_n(i,j) - \Lambda_n(i+1,j)\right|$ and $d_n^V(i,j) = \left|\Lambda_n(i,j) - \Lambda_n(i,j+1)\right|$.
4. In each block compute the variance of the absolute derivatives:

$$\text{var}_n^H = \frac{\sum_k (d_n^H(k) - \mu_{d_n^H})^2}{400} \quad \text{and} \quad \text{var}_n^V = \frac{\sum_k (d_n^V(k) - \mu_{d_n^V})^2}{400}$$

5. Set up a threshold value $\Gamma_2$ for the variance. If the variance of the frame is below the threshold $\Gamma_2$, the frame is determined to be a non-clutter area:

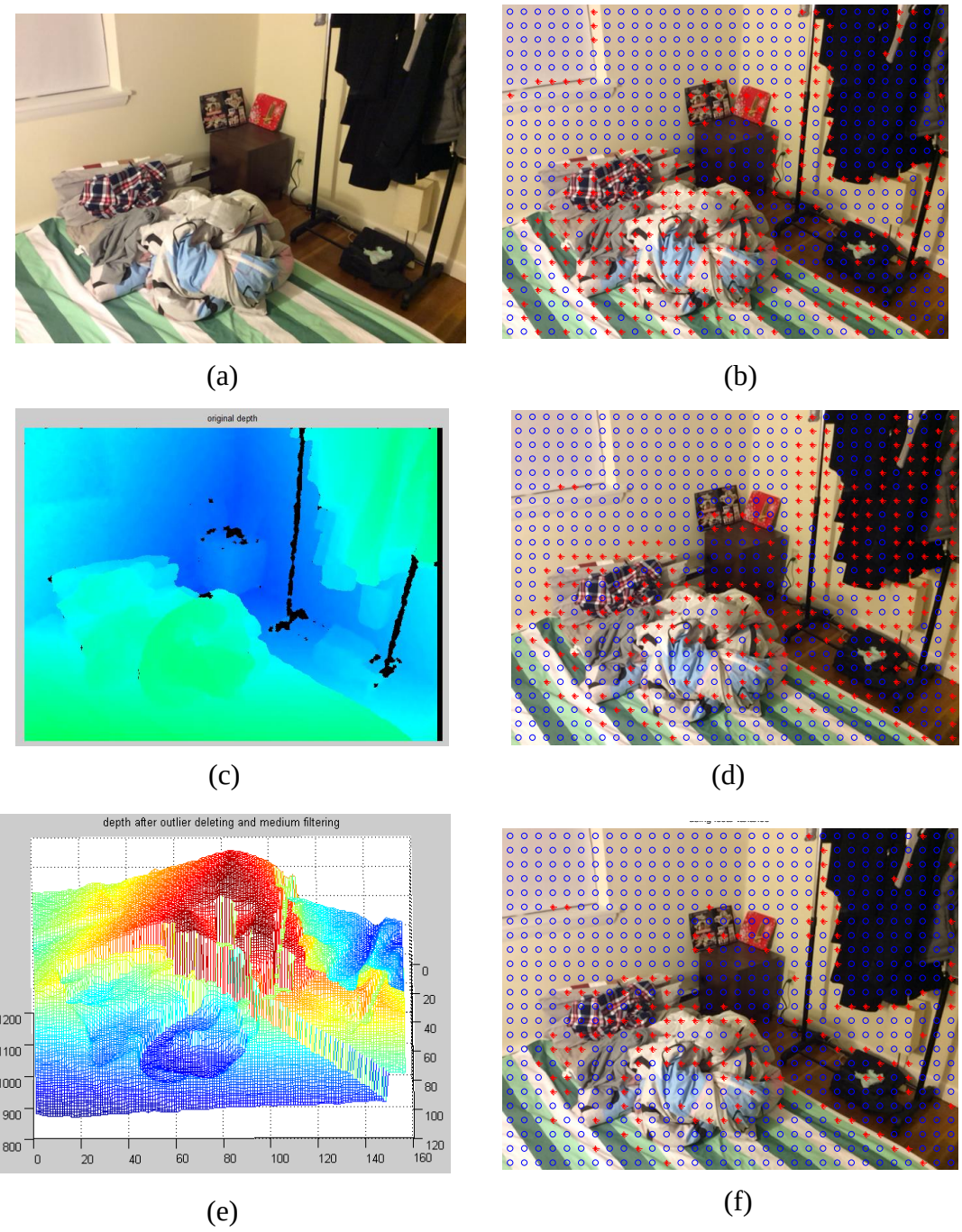
$$\Gamma_2 \begin{matrix} \varepsilon \\ < \\ > \\ \eta \end{matrix} \text{var}_n^H \text{ or } \text{var}_n^V$$

After thresholding, the resulting label matrix $\Im_2$ has a binary value of 1 for clutter or 0 for non-clutter.

6. Combine the 2-D (luminance-based) and the 3-D (depth-based) methods, by dot-matrix multiplication.

## 4.4 Results

The above method is expected to help eliminate false alarms when combined with the 2-D method. Figure 9 shows the results for 2-D method that uses luminance only (Figure 9.b), 3-D method that uses depth only (Figure 9.d), and a fused result by means of "AND" operation (Figure 9.f). As can be seen, many false positives produced by the 2-D method around the window frame, on bed cover and on the wall to the left of the clothing hanging on a rack have been eliminated. This was to be expected since these areas are void of clutter. However, some of the false positives remain (e.g., two red crosses on the horizontal section of the window frame, a number of them at the edge of the bed cover as well as on the wall next to the clothes rack). These errors are partially due to the fact that room surfaces absorb infrared light which then results in invalid depth measurements (black dots in the depth map shown in Figure 9.c – seemingly the clothes rack supports are covered by material that absorbs infrared light). Since few measurements are available in the blocks that overlap these areas, the depth variance calculations are less reliable and lead to some of the errors. As for the horizontal window frame and the edge of bed cover, the false positives occur at discontinuities in the depth surface (window parapet against the wall beneath and the end of bed cover against the floor below). These discontinuities result in an increased depth variance in the corresponding block and indicate a non-smooth surface (which is correct). Although one could increase threshold $\Gamma_2$, this would help in these particular cases but would lead to more misses of clutter (for clutter whose depth structure has only mildly irregular surface).




(a) (b) (c) (d) (e) (f)

*Figure 9 (a) View of a room with low clutter; (b) clutter decisions using the 2-D luminance-based method; (c) depth map from Structure Sensor; (d) clutter decisions using the 3-D depth-based method; (e) depth maps shown as a surface; (f) clutter decisions after fusing results from 2-D and 3-D methods using "AND" operation.*

# 5 Plane fitting to the depth map

As can be seen from the results in the previous section, even the fusion of decisions made by the 2-D luminance-based method and the 3-D depth-variance-based method is not able to eliminate all false positives due to objects of type II (highly-textured planar surface). Since objects of type II are typically paintings, wallpaper, bed covers, etc. one would expect their surfaces to be planar. In the previous method (Section 4), the discovery of object's planarity was attempted by computing local depth variance over a small block (20 by 20 pixels), but this method proved to be erroneous at times. Since of concern are larger planar areas like walls, ceiling, floor, we propose to fit up to 3 planar surface to the depth map with an expectation that a planar fit to a larger area will be more robust than a depth variance measurement within a small depth block. Since type III objects can be easily detected using the 2-D method, the only remaining objects are of type I, some of which may be considered to be clutter (e.g., large boxes), but we leave this to future work.

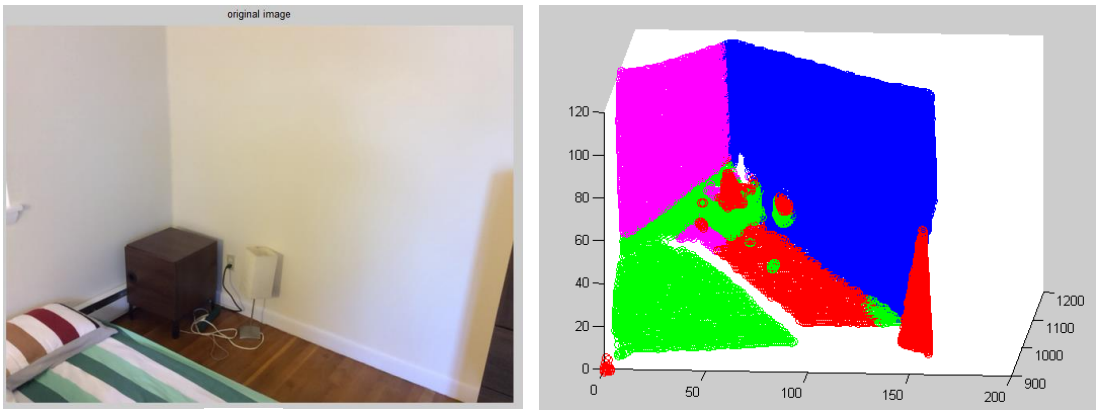
We perform plane fitting (estimation of plane orientation) as follows. First, we find the area in the depth map corresponding to the intersection of two side walls (corner of a room) that is assumed visible in the field of view of the Structure Sensor. We start by finding the largest and smallest values in the depth map, $d_{\max}$ and $d_{\min}$, respectively. Then, we find all depth values that are larger than 90% of the depth range (room's corner is assumed to be the farthest room area from the sensor):

$$d_k \in IA \quad \text{if} \quad d_k \geq d_{\max} - 0.1*(d_{\max} - d_{\min}),$$

where *IA* denotes the intersection area. We then use median filtering to eliminate any isolated depth values in the set *IA*; the remaining points are assumed to be the intersection area of two side walls.

In the second step, we randomly pick a depth value from the wall intersection area: $d_i \in IA$, and find 48 neighboring depth values in a 7 by 7 neighborhood of $d_i$. We use

$d_i$ and its 48 neighbors, denoted as $D_{ini}$, as the starting point for a local RANSAC algorithm to find the first planar fit (blue-colored points in Figure 10.b denoted as $P_{ini}$). This fit is obtained by minimizing the sum of distances from the depth points to the plane (Euclidean distance along the direction orthogonal to the plane). After detecting the first plane, we remove all the points belonging to the first plane and apply the same steps to the remaining points under a constraint that the angle between the already-detected plane and the new plane is as close to 90 degrees as possible. We repeat this procedure for the third plane. After fitting three planes (blue, green, and magenta in Figure 10), we declare the remaining points, which do not belong to any of these planes, as outliers (red colored points). These are likely clutter areas since they do not fit any of the planar surfaces. The knowledge of planar surfaces is expected to help with the elimination of false positives due to type II objects.




(a) (b)

*Figure 10 (a) View of a room from Structure Sensor; (b) The result of plane fitting to the depth map obtained by Structure Sensor in room depicted in (a). The blue, green, and magenta correspond to pixels belonging to a plane (wall or floor) whereas the red corresponds to outlying pixels that did not fit any plane.*

## 5.1 Implementation

Below are detailed steps of our implementation:

1. Find the intersection area of two side walls by identifying all locations in the depth map whose depth exceeds 90% of the depth range: $d_k \geq d_{\max} - 0.1*(d_{\max} - d_{\min})$.
2. Refine the found locations by eliminating isolated points via 3-by-3 median filtering.

3. Pick a random point in the intersection area and then pick another point 20 pixels away horizontally from the first point, and use the latter point as the starting point. Find 48 nearest neighbors of the starting point in a 7-by-7 neighborhood.
4. Estimate the location of the plane using the starting point and its 48 neighbors (49 points in total): the plane passes through the centroid of these 49 points.
5. Estimate the orientation of the plane by minimizing the sum of orthogonal distances of these 49 points to the plane. This step is carried out by constructing a 3 by 3 covariance matrix of these 49 points and performing eigendecomposition of this matrix.
6. Set up a threshold value t (e.g., t = 9): any point whose orthogonal distance to the plane is smaller than t is considered to be the point belonging to this plane: $d_j \in P_{ini}$ if $dist(d_j, P_{ini}) < t$
7. Set up a threshold value $N$ (e.g., $N$ = 1800) : plane $P_{ini}$ is considered as a potential plane if there are at least $N$ points belonging to the plane (after thresholding is step 6). If the number of points belonging to $P_{ini}$ is less than $N$, repeat starting from step 3 until the number of points exceeds $N$.
8. Refit a new plane $P_{new}$ using all points that are within the distance $t$ from $P_{ini}$ using the method from step 5.
9. Repeat steps 6-8 ten times. The best-fitting plane is the one with the smallest average orthogonal distance from depth points to the plane. Denote the set of these points by $\psi_1$
10. Repeat steps 3-9 ten times to obtain ten sets of points: $\psi_1$, …, $\psi_{10}$, and compute their union: $\psi_{tot} = \psi_1 \cup \psi_2 \cup \ldots \cup \psi_{10}$.
11. Compute the average orthogonal distance from points in $\psi_{tot}$ to the ten planes calculated in step 10. The best plane is the plane with the smallest average orthogonal distance:

$$P_{opt} : \text{argmin}_P \sum_{d_j \in \psi_{tot}} orthogonal\_dist(d_j, P)$$

12. Remove all the points that fit the first plane, and fit the second plane using the remaining points. Repeat for the third plane. An extra constraint is added to the second and third plane: the second plane is limited to be as close as possible to being perpendicular to the first plane. Similarly, the third plane is constrained to be as perpendicular as possible to the first and second plane.

## 5.2 Results

Figures on the following pages show the results of the plane fitting method, from the simplest scenario to more challenging ones. The points associated with blue, green, and magenta are the points belonging to three planes. The points associated with red are the outliers that do not belong to any plane.

As can be seen in Figure 11 the plane fitting approach works quite well. The walls and ceiling in the empty room are accurately estimated. Similarly, the two walls in the furnished room are quite well estimated while the floor (green) is not as accurate due to the presence of the bed and a cube-shaped table in the corner. In both cases, as expected, very little clutter is detected (the lamp in Figure 1Figure 11.d, otherwise false positives). In the case of slightly-cluttered room (Figure 11.f), the walls are accurately estimated however the fit to the floor surface is very poor. This should not be surprising since there are several horizontal planar surfaces (bed, table, floor) so the algorithm has difficulty with making a decision.

Figure 12 and Figure 13 show two different runs of the plane fitting algorithm on the same depth maps. While the results in Figure 12 are quite successful, those in Figure 13 are not. Figure 12.b, shows clutter labels obtained by the 2-D method (thresholding of the luminance gradient magnitude). Clearly, there are many false positives on the bed cover, some on window frame and some on the wall around the clothes rack. Figure 12.d shows results for the plane fitting method which produces no false positives on the bed. As can be seen in Figure 12.e, the intersection of the two results (fusion by means of “AND” operator) results in clutter detections in the correct area (bundle of clothing on the bed) and a few detections at the boundaries of clothes rack and the clothing. The latter detections can be considered as false positives if one considers the clothing rack as non-clutter.

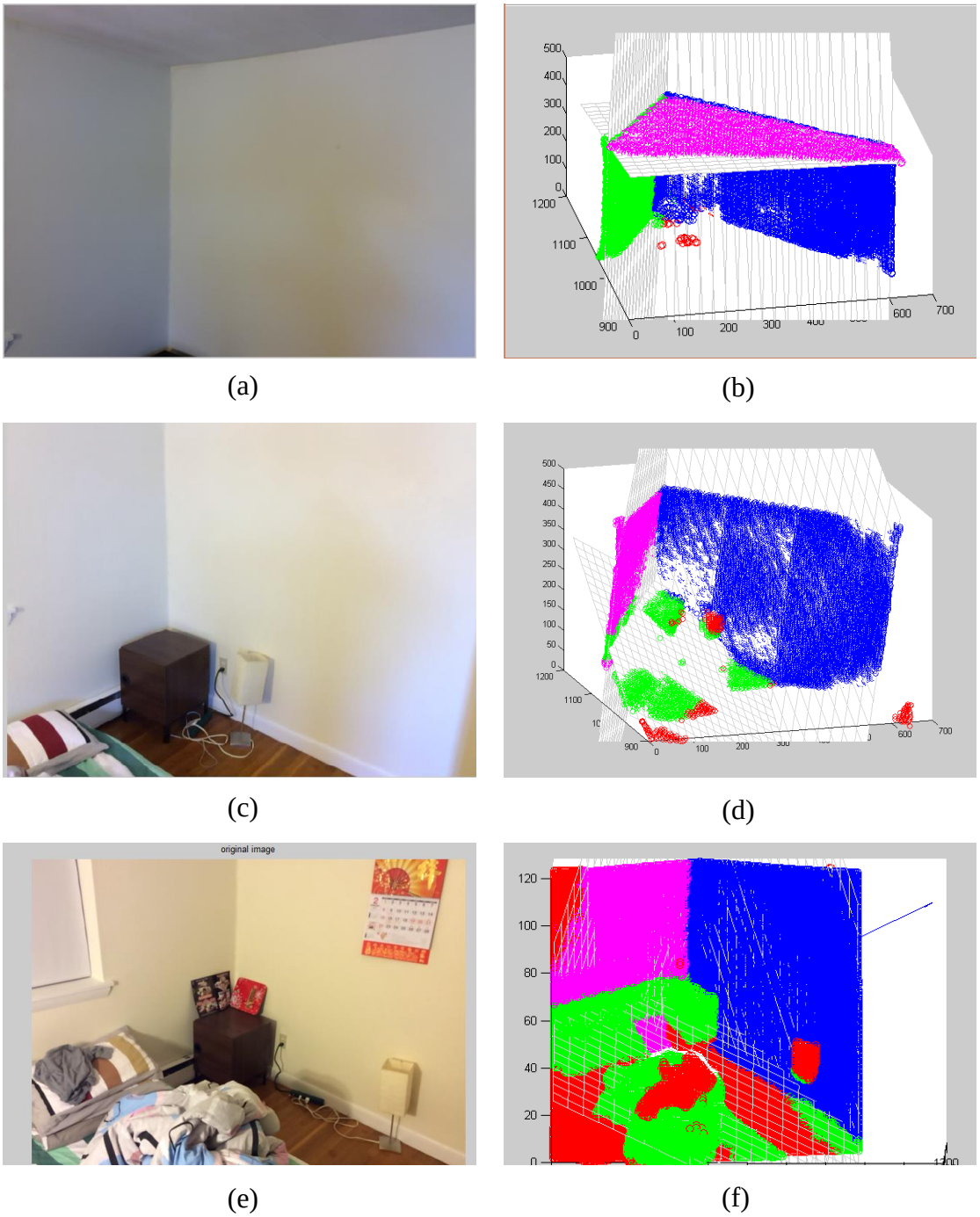



*Figure 11 (a) Empty room scenario and (b) corresponding plane estimates; (c) furnished room scenario and (d) corresponding plane estimates; (e) slightly cluttered room scenario and (f) corresponding plane estimates.*

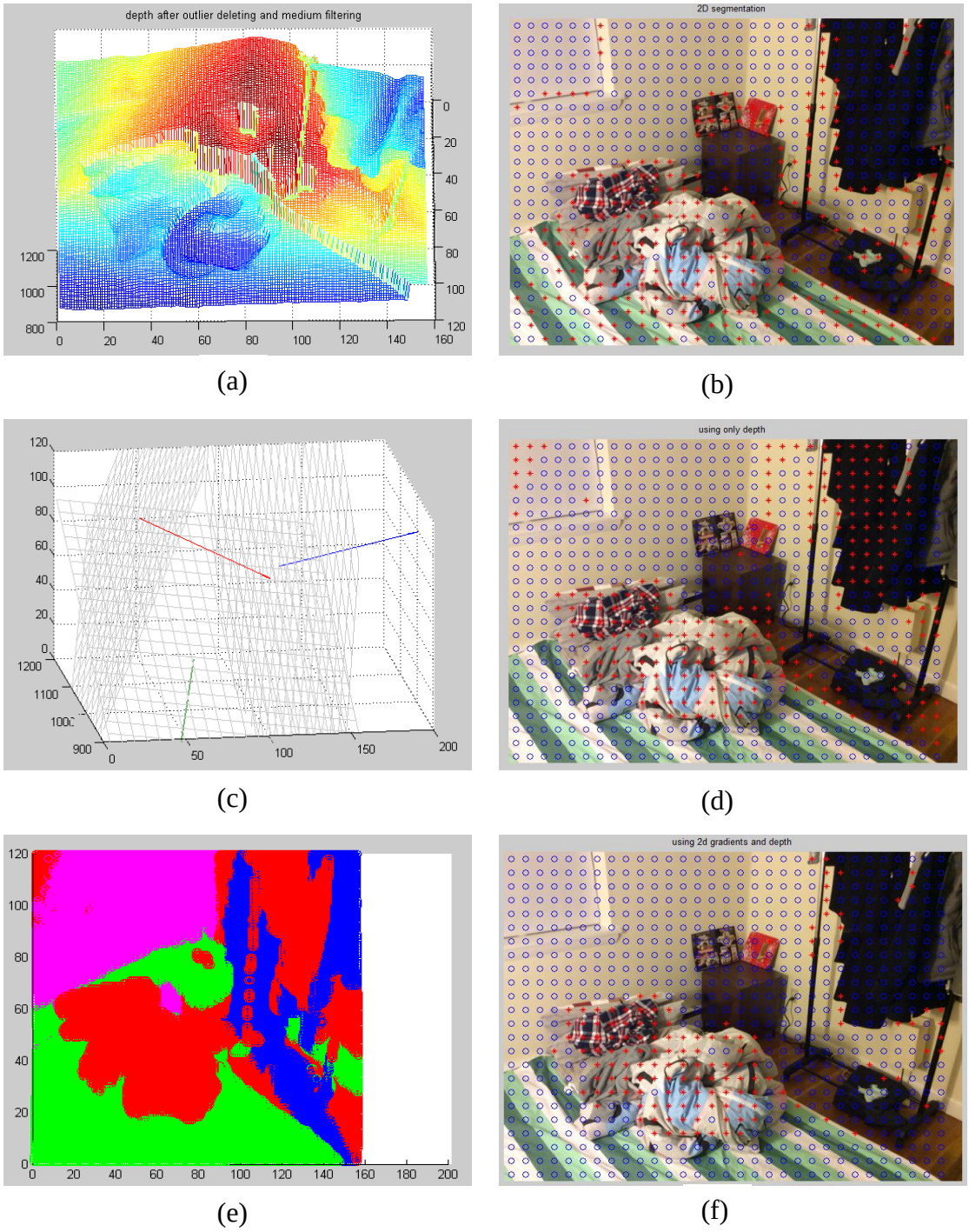



*Figure 12 (a) Depth map from Structure Sensor as a surface plot; (b) clutter labels from the 2-D method (thresholding of luminance gradient magnitude; (c) three fitted planes as a surface plot; (d) clutter labels from the plane fitting method (outliers are considered to constitute clutter); (e) plane detection results; (f) fusion of results from (b) and (d) by means of "AND" operator.*

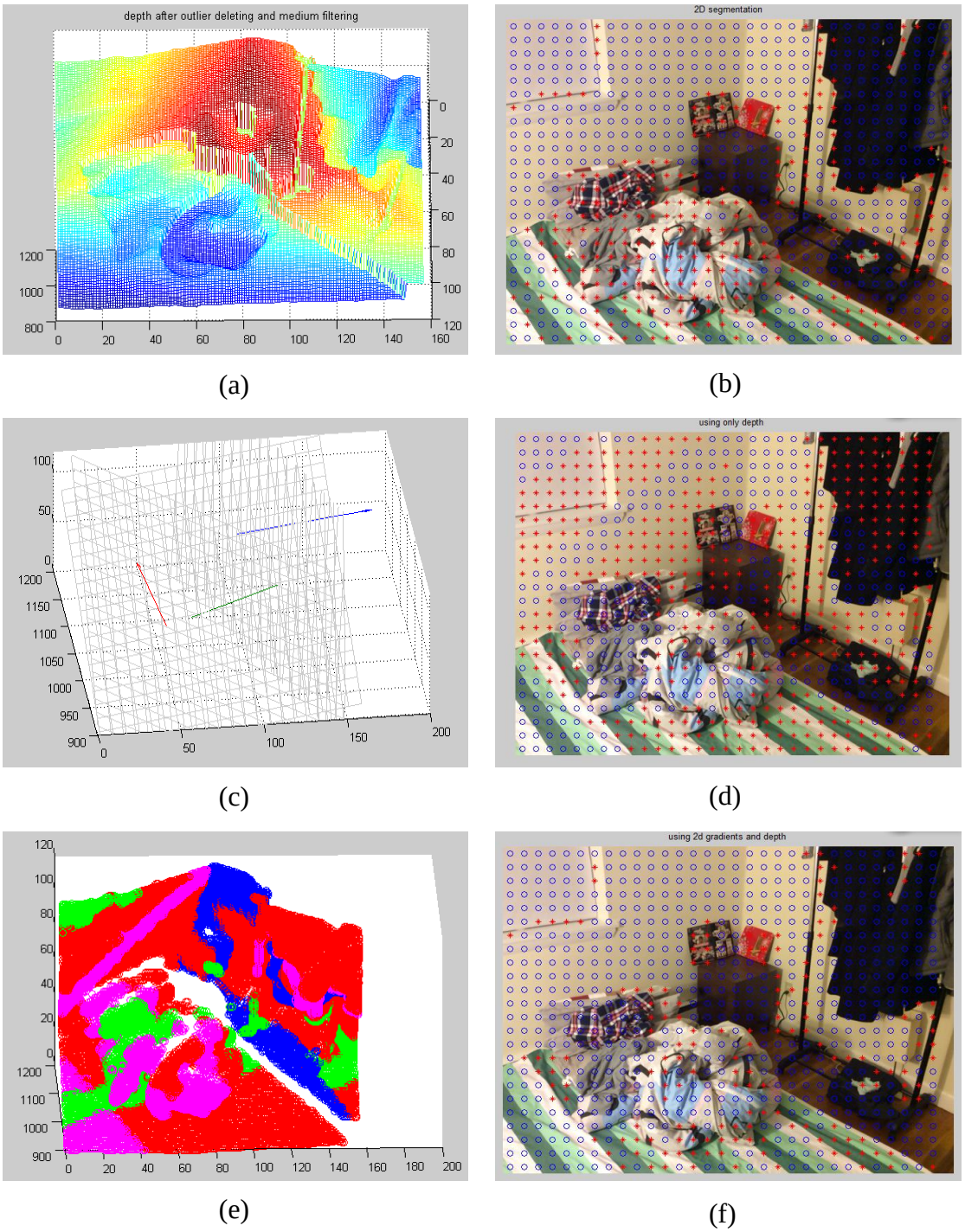



*Figure 13 (a) Depth map from Structure Sensor as a surface plot; (b) clutter labels from the 2-D method (thresholding of luminance gradient magnitude; (c) three fitted planes as a surface plot; (d) clutter labels from the plane fitting method (outliers are considered to constitute clutter); (e) plane detection results; (f) fusion of results from (b) and (d) by means of "AND" operator.*

As for the results in Figure 13, they are not successful. The planes of the walls and the bed were not correctly detected, and therefore most of the bed surface is deemed as clutter in Figure 13.d and consequently some of these false positives remain in the fused label field in Figure 13.f. Similarly, the wall with the window was not detected and thus the false positives around the window frame remain in the fused result in Figure 13.f. The failure of the plane fitting method in this case is due to the random initialization.

# 6 Conclusions

In this project, we attempted to detect areas of clutter in a room based on a captured luminance image and depth map. We assumed that clutter corresponds to a high incidence of luminance edges and we proposed a 2-D method that thresholds the average magnitude of luminance gradient in 20-by-20 blocks. The method performed reasonably well but resulted in excessively many outliers, especially in textured but planar areas (e.g., wallpaper). In order to deal with such outliers one needs to distinguish areas with dense edges but underlying planar 3-D surface (e.g., wallpaper) versus similarly detailed areas that have a complex underlying 3-D structure (clutter). To this end we have employed Structure Sensor that captures depth map in its field of view, and developed one local (block-based) and one global planarity detection method. The local method that simply thresholds the variance of the magnitude of horizontal and vertical depth derivatives within 20-by-20 blocks, when combined with the 2-D method, improves the accuracy of clutter detection but is not very robust as it uses data within a small block only. The global method that fits 3 mutually orthogonal planes to the captured depth maps in order to establish planarity of the walls and floor/ceiling, when combined with the 2-D method, works better than the local method. However, it is prone to catastrophic failures due to random initialization of the plane fitting algorithm. For higher-complexity room images (more clutter and furniture), the plane fitting algorithm is likely to fail when detecting planes thus very likely severely affecting final results. Clearly, a different approach is needed to assure robust performance in such a scenario.

# 7 References

[1] Frost, R. O., Steketee, G., & Grisham, J. (2004). Measurement of compulsive hoarding: Saving inventory-revised. *Behaviour Research and Therapy, 42*, 1163–1182.

[2] Lervolino, A.C., Perroud, N., Fullana, M.A., et al. (2009). Prevalence and heritability of compulsive hoarding: A twin study. *American Journal of Psychiatry, 166*, 1156–1161

[3] Frost, R.O., Tolin, D.F., & Maltby, N. (2010). Insight-related challenges in the treatment of hoarding. *Cognitive and Behavioral Practice, 17*, 404–413.

[4] Frost, R., Steketee, G., Tolin, D., & Renaud, D. (2008). Development and validation of the Clutter Image Rating. *Journal of Psychopathology and Behavioral Assessment, 30*, 193–203.

[5] Tolin, D., Frost, R., & Steketee, G. (2010). A brief interview for assessing compulsive hoarding: The Hoarding Rating Scale. *Psychiatry Research, 178*, 147–152.

[6] Tolin, D. F., Frost, R.O., Steketee, G., Fitch, K.E. (2008). Family burden of compulsive hoarding: results of an internet survey. *Behaviour Research and Therapy, 46*(3), 334-344.

[7] Samborski, A.M. (2014). Clutter Image Rating Application. Poster presentation at 2014 UROP Symposium, Boston University (Advisor: Prof. J. Muroff).

[8] http://structure.io