# On the Analysis of 3D Room Layout from 2D Images

Rob J. Dallara
Department of Computer Science,
University of North Carolina at Chapel Hill
dallara@live.unc.edu

## I. Introduction

One of the most important problems in the field of robotics is robot navigation in complex, ever-changing 3D environments. However, many robots in today's world still lack visual cameras and sensors which can help guide them through such environments without any collisions with obstacles. Such robots must therefore rely solely on 3D information about their environments which can only be obtained from 2D images taken by their visual cameras and sensors. The remaining problem is to figure out an effective, efficient way for a robot to obtain accurate 3D information about its surroundings from a single, two-dimensional RGB-D image.

## II. Related Work

This project is based on the *Parsing Indoor Scenes Using RGB-D Imagery* paper written by Camillo J. Taylor and Anthony Cowley [5]. However, there are a number of other papers and publications which have also looked at this problem and similar problems stemming from it. For example, Emmanuel Perez Bonnal published in his July 2011 Master's Thesis [1] some possible means of allowing a robot to reconstruct 3D rooms and even larger maps based on RGB-D Kinect camera images. A method for using dynamic programming to reconstruct building interiors from single images is also presented in *A Dynamic Programming Approach to Reconstructing Building Interiors* [2]. In addition, the *Reconstructing Building Interiors from Images* [3] paper proposes an automated method for parsing architectural scenes of building interiors to gain 3D information. Another paper worth noting is the *Indoor Scene Segmentation using a Structured Light Sensor* [4] paper which also looks into an algorithm for obtaining 3D info from 2D indoor images using Microsoft's Kinect camera and its depth measurements.

## III. Problem Definition

The input P to this problem is a two-dimensional, RGB-D image of an interior room inside a building with straight, rectangular walls. This input image P may or may not have a rectangular ceiling near the top of the image or a rectangular floor near the bottom of that same image. In addition, the room in image P might contain additional smaller, everyday objects along with its high-level rectangular structure. The output Q is an image containing solid, color-coded regions representing different rectangular surfaces in the room (i.e. the floor, ceiling, and walls) without any furniture or other objects blocking their view. Any visible ceiling in Q is shaded in red, any visible floor is shaded in orange, while the walls are shaded in random colors.

## IV. Methods

In order for the algorithm to obtain the final results, the input image is first smoothed to eliminate any noise (which could potentially lead to misleading wall pixels later in the algorithm). To do this, the RGB value of each pixel is recalculated based on the average value of itself and all of its neighbors [6]. After this, the image is parsed one pixel at a time from the top left corner to the bottom right corner in order to look for candidate wall edge pixels. In order to determine if a pixel is an edge pixel, the RGB values of each pixel are converted into HSV (hue-saturation-value) triplets, and two Sobel edge filters [6] are used to “weigh” the brightness (“value”) of the current pixel and its neighbors. If this weight exceeds a certain threshold for one of the filters, the algorithm flags the current pixel as a wall edge pixel and colors it blue. An additional method which can be used to label the wall edges allows the algorithm to keep track of the changes in the brightness value from pixel to pixel. If the change in brightness from the previous pixel to the current one exceeds a certain threshold, the algorithm will label the current pixel as a wall edge pixel.

Next, the algorithm smooths all of the wall edge data in order to eliminate any noise which might occur within it. In order to do this, the algorithm iterates over each labeled wall edge pixel and analyses its surroundings.

The number of edge pixels within a certain distance of the current wall edge pixels is obtained, and this along with the known size of the analysis area is used to calculate the percentage of the current and nearby pixels that are also wall pixels. If this percentage is below a certain threshold, the current wall edge pixel is classified as "noise" and is restored to its original smoothed value.

After this step is complete, the algorithm attempts to create solid lines corresponding to different wall edges in the input 2D image. In order to do this, the algorithm repeatedly selects a random wall edge pixel to run a linear regression on. The algorithm looks at a narrow, horizontal as well as a narrow, vertical region surrounding the current edge pixel and records the number of edge pixels in each of those regions. If more edge pixels are found in the horizontal region fir the current pixel, the linear regression is calculated based on the horizontal region pixels. Otherwise, the linear regression involving the current edge pixel is calculated based on the vertical region pixels. All of this is done to increase the likelihood that the calculated linear regressions more closely fit the different walls found during earlier analysis. Finally, a solid line corresponding to the chosen linear regression is drawn in the image, and any wall edge pixels which it passes over are eliminated from future analysis. This is repeated until all of the wall pixels are eliminated from the analyzed room image.

In the last step of the algorithm (implemented separately due to inaccurate linear regressions), the different planes formed by the solid edges are classified into wall, floor, and ceiling regions. In order to do this, each region is first color-coded using a randomly chosen, non-black color. This is done through a pass over all of the pixels in the image from the top-left corner to the bottom-right corner. Every time a black pixel is encountered, it is labeled with a random color, and its surrounding region (delineated by the solid wall lines calculated earlier) is also labeled with that same color. Finally, the algorithm looks for any regions which might correspond to floor and ceiling regions (and colors them in accordingly if they are suspected). This is done by analyzing the number of visible regions on the top and bottom of the image and operation under the assumption that any floor or ceiling plane will be the middle region in a chain of visible regions at the top or bottom of the image. This algorithm also assumes that any floor or ceiling region is only visible if there are an odd number of regions running along its respective area of the image. For example, an image with two regions at the bottom is assumed to have no visible floor plane, as these two regions are most likely different walls instead of possible floor planes.

## V. Results

While some of the final results are actually fairly accurate, many of them fail to catch more subtle walls in some of the input room images. In addition, the algorithm also gets fooled by certain image features such as shadows.

The algorithm manages to delineate many of the walls in some of the images fairly accurately. For example, the image in figure 1 is labeled with wall edge pixels in figure 2 very accurately in most locations.

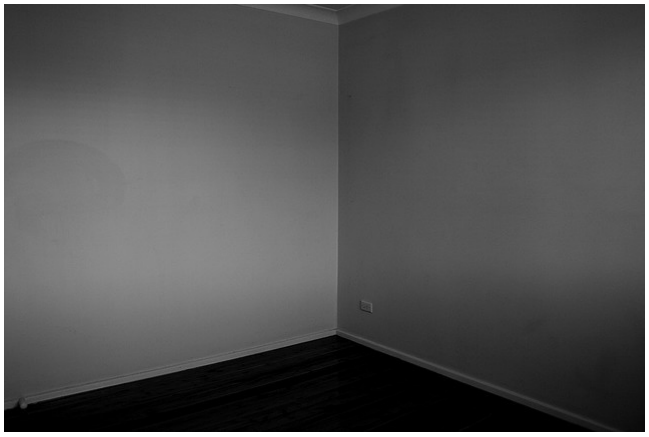
**Figure 1: A simple image containing two walls and a floor Taken from Flickr photo stream of "Evan Watson"**

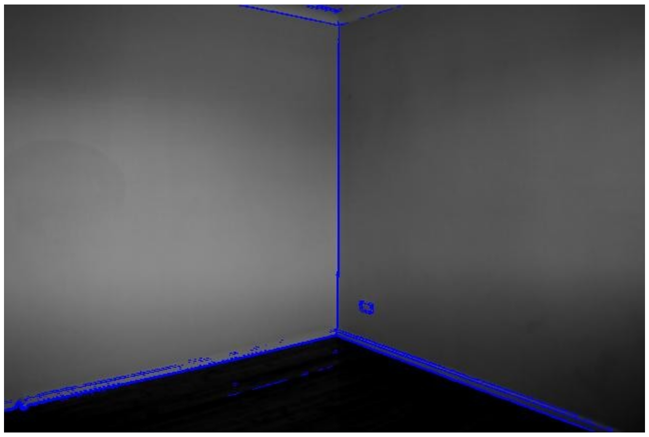
**Figure 2: Labeled wall edges for image in figure 1.**

Another sample image and its corresponding wall edge labeling can be found in figures 3-4 below.

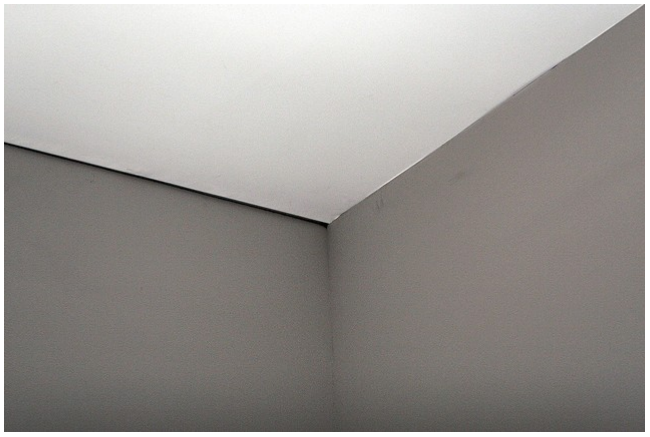

**Figure 3: The upper corner of a room**
**Taken from Flickr photo stream of "Pulpolux !!!"**

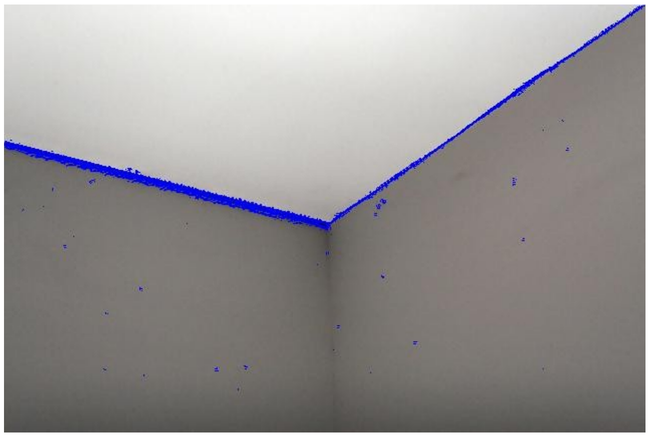

**Figure 4: Labeled wall edges for image in figure 3**

On the other hand, the algorithm  fails to accurately label many wall edge pixels in the figure 6 image based on figure 5 below.

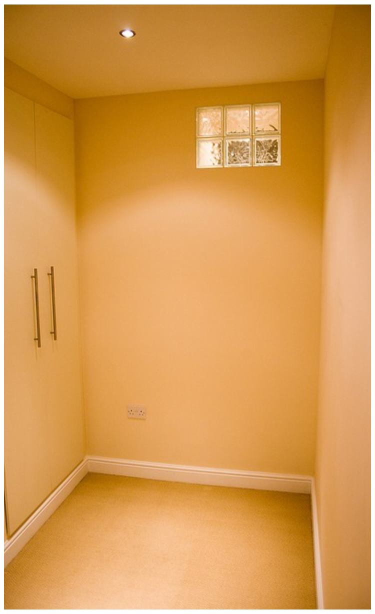

**Figure 5: An empty room with a light and window**
**Taken from Flickr photo stream of "YayAdrian"**

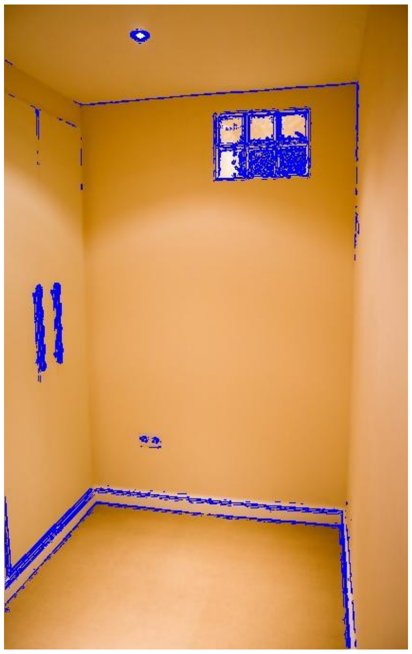

**Figure 6: Labeled walls for image in figure 5**

As the above sets of images reveal, the algorithm can easily get fooled by shadows, wall features, windows, and more subtle wall boundaries. In addition, even an accurate wall edge pixel labeling can often give rise to linear regressions which do not accurately reflect the different wall edge structures. This is due to the fact that the algorithm only looks ahead from different wall edge pixels by a short distance to look for pixels to add to each linear regression. This linear regression is used to infer the best-fit line for an entire wall edge even though it only takes into account a small percentage of the total number of the complete set of pixels for each wall boundary. This means that any slight pixel anomalies in the regression point set will add up dramatically to produce inaccurate lines reflecting the orientations of different wall edges. Examples of this problem in action can be found in the images below.

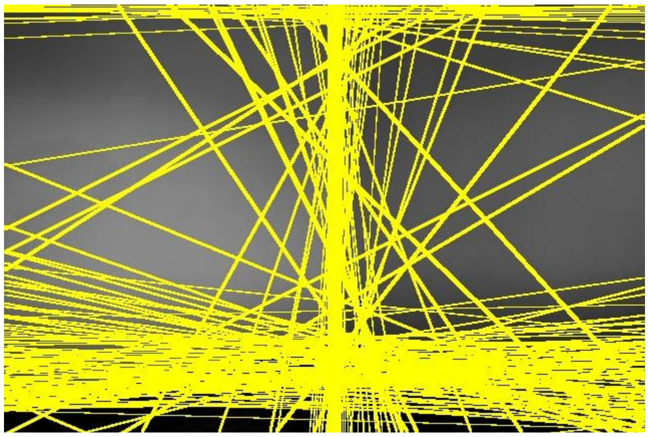

**Figure 7: Noisy, chaotic linear regressions for image in figure 1**

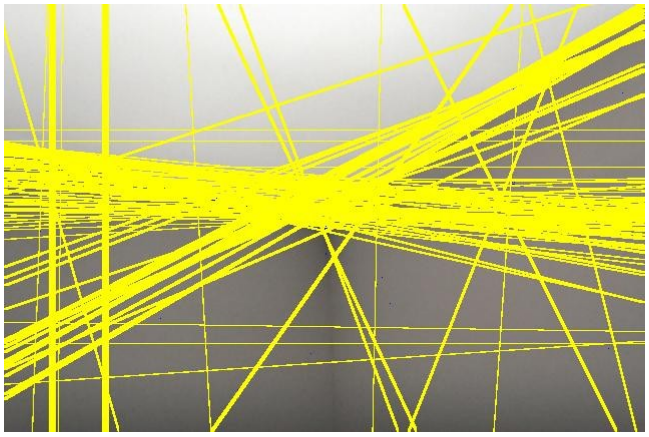
**Figure 8: Noisy, chaotic linear regressions for image in figure 3**

As a result of the mostly inaccurate linear regressions in the algorithm, the region-labeling part of the algorithm had to be tested separately from everything else. In order to do this, a series of "wire frame" images were created based on the input images for the algorithm. Some examples of these wire frame images can be found in figures 9 and 10 below.

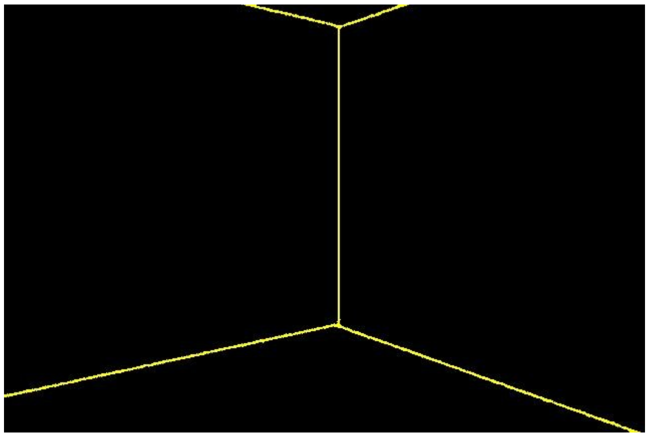
**Figure 9: Wire frame image based on image in figure 1**

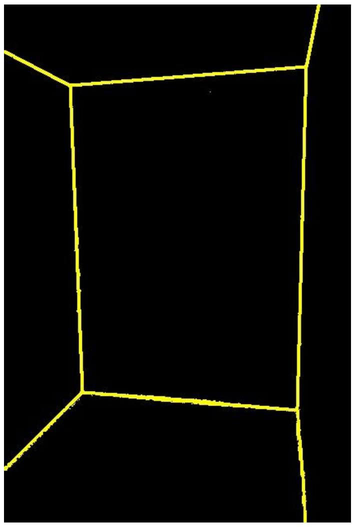
**Figure 10: Wire frame image based on image in figure 5**

The algorithm then labeled the different regions of these wire frame images as floor, ceiling, or wall regions based on several assumptions on its particular data set. First, any ceiling plane will almost certainly be found towards the top of the image of a room, and any floor plane will most likely be found at the bottom of the image. Any other planar regions in the input image are assumed to be walls. In addition, when an even number of planes are found at the top or bottom of an image, they are all assumed to be wall planes with no floor or ceiling visible. Otherwise, the ceiling or floor region is assumed to be the middle region on the top or bottom of the image respectively. Some sample results can be found in the figures below.

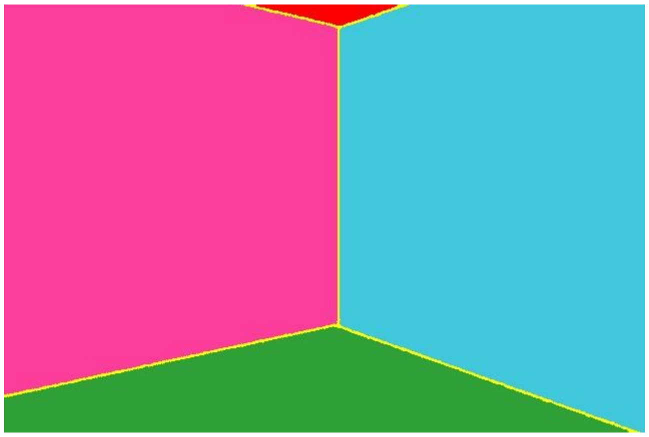
**Figure 11: Semi-accurate regions for image in figure 1**

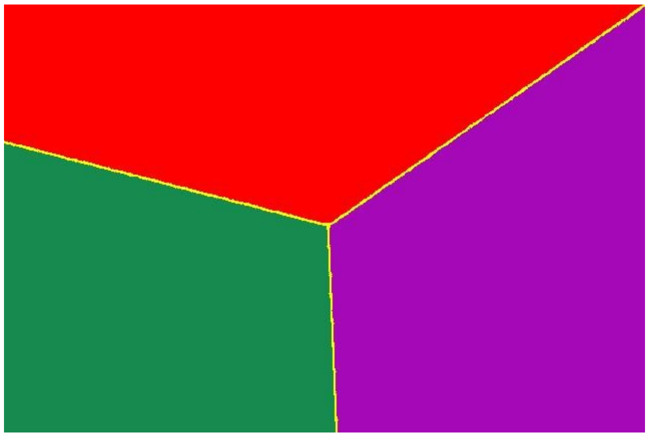
**Figure 12: Accurately labeled regions for image in figure 3**

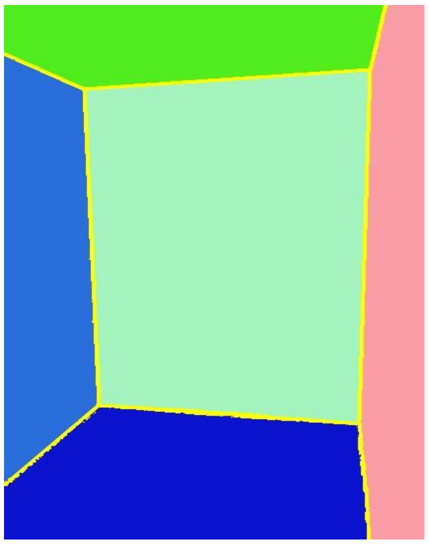
**Figure 13: Inaccurately labeled regions for image in figure 5**

A key aspect of the above three figures is that any predicted ceiling planes are colored red, while predicted floor planes are colored orange. Regions assumed to be walls are labeled using random colors.

## VI. Conclusion and Future Work

While the algorithm discussed does a reasonable job of outlining different walls in a room, it is still not perfect and needs improvements in order to deal with subtleties such as faint wall edges, small wall objects (such as electrical outlets), and shadows. In addition, the method of linear regression for labeled wall pixels needs to be improved or replaced in order for the algorithm to obtain better solid wall edge outlines. Finally, the algorithm for region labeling needs to be improved so that it more accurately labels floors and ceilings even with image noise or other subtleties (such as those found in the image of figure 13). This could be done through region classification heuristics or perhaps through other means such as probabilistic methods or reinforcement learning using both training and testing images. Once these improvements are made, the region-labeling procedure could also be integrated with the rest of the algorithm presented in this paper. This would almost certainly be the most interesting next step for anyone choosing to further improve this algorithm in a future continuation of this project.

## VII. Acknowledgements

I would like to thank professor Ron Alterovitz for his feedback on my input data and progress report for this project. His recommendations for the simplicity of my input data in particular made this project more feasible and interesting to implement.

## References

[1] E. Perez Bonnal, "3D Mapping of indoor environments using RGB-D Kinect camera for robotic mobile application," Universitat Politècnica de Catalunya Politecnico di Torino, July 2011

[2] A. Flint, C. Mei, D. Murray, I. Reid, "A Dynamic Programming Approach to Reconstructing Building Interiors." In *Computer Vision–ECCV 2010*, 2010

[3] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Reconstructing Building Interiors from Images." In *Computer Vision, 2009*, 2009

[4] N. Silberman and R. Fergus, "Indoor Scene Segmentation using a Structured Light Sensor." In *Computer Vision Workshops (ICCV Workshops)*, 2011

[5] C. J. Taylor and A. Cowley, "Parsing Indoor Scenes Using RGB-D Imagery." In *Robotics: Science and Systems Conference 2012*, 2012

[6] Berg, Tamara. "Vision1." *Artificial Intelligence*. N.p., n.d. Web. 01 May 2014. Found on *http://tamaraberg.com/teaching/Spring_14/*