1 Introduction

There has been a lot of work recently in the area of mobile computer vision. Various computer vision algorithms have been implemented on portable devices such as mobile phones to perform complex tasks such as object recognition, object tracking, image matching, scene understanding, etc. There has however not been a lot of work done towards measuring a room or a scene using a mobile phone. More precisely, image understanding has not been extended towards measuring objects in the scene or the dimensions of the scene itself. We believe that there is a lot of untapped potential here, particularly because of the numerous applications of this project. This app can be used by physically challenged people for navigation in a wheelchair. Physically challenged people would like to have the ability to navigate and move seamlessly while on a wheelchair. Modern electronic wheelchairs have a lot of controls on them, hence making them cumbersome for the people who use them. Instead if the person was able to select where he or she wanted to go on the phone, the phone could then return the coordinates of that location, making it easier to provide input to the wheelchair. Augmented reality has gained a lot of popularity recently. We believe that our app can also be used in such applications. For example, playing games like squash, ping-pong etc between two android phones. When parking space is hard to find, under such cramped conditions, it becomes even more hard to judge whether your car can fit between two cars. A person can use this app to determine whether there is sufficient space to park between two cars in a parking lot. Another novel application is that of reconstructing a 3D model of the environment for applications such as virtual tours, etc. This app can also be used to measure the room and furniture inside it. For example, if you have just moved into a house or want to get some new furniture, you may not be sure of how much space is available. Using this app, you can measure the room up to scale, and then at the store measure the furniture up to the same scale and decide if it will fit nicely or not.

A lot of work has been done in the area of single image based scene understanding. Saxena et. al [1],[2] used Markov Random Fields to identify the orientation and 3D positions of super pixels in the image and then reconstructed a 3D model from it. A lot of work has also been done by Hebert et. al [3], [4], [5] in interpreting a scene from a single image. They generate a coarse geometric model of the image which divides the image into three regions, namely ground, sky and surfaces. They also use qualitative descriptions based on blocks to interpret the scene. Hoiem et. al [6] generated a box layout of indoor scenes with clutter. However, most of these approaches seem to require extensive training and are also computationally expensive. For developing on mobile platforms, we require algorithms that can run in real time and also have low latency. Another important constraint is the bandwidth available for data to move between server and client. Hence we propose a simple method which exploits geometric properties of the scene, and does not require training. Further, we wish to encourage user interaction with the algorithm, which we believe will enhance the user experience, and also provide solutions which are user friendly. In the next section, we explain our method, with details on the underlying mathematics, and also provide some preliminary results.
2 Technical Details

Up to now, we have successfully formulated our mathematics for reconstructing the 3D environment. We simulated our ideas in MATLAB to make sure that the algorithm works properly. Our next step is to familiarize ourselves with the Android developing environment and its libraries for using various parts of the phone, such as touch screen, camera, keypad. Furthermore, we need to set up a server that could deal with two clients at a time. Below, we describe our mathematics and our simulations in the rest of this section.

2.1 3D Reconstructing the Environment

We have made the following assumptions about the environment and the camera:

1. Indoor environments consist mainly of a flat floor and gravitationally vertical straight walls.
2. The images are obtained by perspective projection of a calibrated camera with the calibration matrix \( K \).
3. The vertical axis of the camera is roughly perpendicular to the floor plane.

For any image which follows assumptions 1 and 2, we can reconstruct the 3D location of any point in the frame under its coordinate system, if the boundary lines between the walls and the floor in the frame are known.

First, we set the camera center as the origin of the coordinate system. Based on the geometry of perspective projection, the 3d location \( \mathbf{P}_i \) of a point at position \( \mathbf{p}_i \) in the image plane must satisfy:

\[
\mathbf{P}_i = \alpha_i \mathbf{K}^{-1} \mathbf{p}_i = \alpha_i \hat{\mathbf{p}}_i
\]

for some \( \alpha_i \). Thus, \( \mathbf{P}_i \) is restricted to a specific line passing through the origin. Now, if this point lies on the floor plane with normal vector \( \mathbf{n}_{\text{floor}} \), the exact 3D location could be determined by the intersection of the line from the perspective geometry and the floor plane:

\[
d_{\text{floor}} = -\mathbf{n}_{\text{floor}} \cdot \mathbf{P}_i = -\alpha_i \mathbf{n}_{\text{floor}} \cdot \hat{\mathbf{p}}_i
\]

where \( d_i \) denotes the distance of the camera to the floor. Thus, we could determine the 3d location of any points that lies on the floor plane. By setting the floor plane parallel to the x-z plane, and the distance \( d_{\text{floor}} \) to the origin as the height of the camera, \( h_{\text{camera}} \), equation 2 become simpler. For any points on the floor plane with location \( \hat{\mathbf{p}} = (\hat{u}_i \hat{v}_i 1)^T \) on the calibrated image plane,

\[
\mathbf{P}_i = \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = h_{\text{camera}} \begin{pmatrix} \frac{\hat{u}_i}{\hat{v}_i} \\ 1 \end{pmatrix}
\]

Next, we can determine the normal vector of a wall \( \mathbf{n}_{\text{wall}} \) based on a given boundary line, using the vertical wall restriction from Assumption 1:

\[
\mathbf{n}_{\text{wall}} = \mathbf{n}_{\text{floor}} \times \mathbf{v}_{\text{boundary}}
\]

where \( \mathbf{v}_{\text{boundary}} \) is the direction vector of the boundary line in 3d. If a point lies on the wall plane with position \( \mathbf{p}_j \) in the image frame, its 3d location is the intersection of the line from the perspective geometry and the wall plane

\[
d_{\text{wall}} = \alpha_j \mathbf{n}_{\text{wall}} \cdot \hat{\mathbf{p}}_j
\]

Based on the geometry above, we can reconstruct any set of points in their 3D location.
2.2 The System and Client-Server Design

There are three main processes in our system as follows:

1. Identify the boundary of the walls and the floor plane.
2. Reconstruct 3D structures of the environment
3. Extract feature correspondences and their 3D relationships between images from two users.

We are planning to do the first step in the server side. That is, the user takes a picture and sends it to server and the server returns the boundaries of the floor plane which is represented by a set of lines. In our implementation, a line will be represented by its two end points. Once the information of the floor plane is ready, all the computation for 3D reconstruction is simple and fast. To ensure real time interaction with the user, the second part of the system is going to be computed at the client side. For the last part of the system, we need to use the server to provide a connection between the two users. However, this is still a one-time process. Both users should send their images to the server side to do feature matching and further obtain the spatial relationships between the users. Once the relationship is established, all the information between the two users could be computed on the phone.
3 Milestones Completed

General Milestones

- Literature survey.

Milestones in Server Side

- Sending/Receiving image to/from the client.
- MATLAB Simulation of our algorithm for obtaining the floor plane in a given image.

Milestones in Client Side

- Taking a picture with the camera.
- Sending/Receiving Image to/from the server.
- Formulation and MATLAB simulation of all our 3D reconstruction algorithm.

4 Milestones Left

General Milestones

- Testing and validation of system.

Milestones in Server Side

- Sending/Receiving images/data to/from two clients. (Nov. 20)
- Feature matching between two images and computation of the relative pose. (Nov. 25)
- Converting our MATLAB code for obtaining the floor plane into java or c/c++. (Nov. 25)

Milestones in Client Side

- User Interface development. (Nov. 25)
- implement our MATLAB code for 3D reconstruction into the phone. (Dec. 3)

References


