2DTO3D : Designing 3D Models from 2D Images

Harsha.K.C
harsha.kc@iiitb.org
MT2013056

December 2014

1 Introduction

In order to design a 3D model one should have a prior knowledge of CAD software which are slightly complex unlike the image editors. This becomes a potential drawback if one has to use 3D printing and should rely on only the available CAD models. If there could be a system which would allow the user to use an image and derive an approximate CAD model of the same, this barrier of being a novice in design can be broken. In essence this involves approximating the unseen depth of the object from an image. The process of projecting from 3D to 2D is simpler but the reverse process is not straight forward.

A photograph of a model has hidden depth information which cannot be computed without making reasonable assumptions. The photo of a cuboid might look like a cube depending on the camera position and orientation. The challenge with Computer Vision is we need at least two photographs to compute this depth information. There are various approaches as well to name a popular one, called Structure From Motion(SFM) which makes use of multiple images for scene/object re-construction. The Human eye can perceive depth due to its stereoscopic nature. If we assist the computer with Human interaction in building these models, one can clearly solve the problem of depth perception which is not straight forward to a machine.

2 Objective

The objective of this project is to generate a 3D model of an object or thing, given an image. The 3D model/mesh could be further enhanced in any CAD system. The 3D model generation will be assisted by minimal user interaction.

The Motivation for building a tool like this was to aid modelling concerns for 3D Printing. Naive users who don't know how to use a CAD system are limited with the designs which can be printed. Say a household item was misplaced, a door knob broke, a cover of the remote was lost, one can print these objects provided there are models. Or can design one themselves using images. The inspiration for such a tool came from the 3Sweep[1] technique.

3 Project Scope

The aspects which lie in the scope of this projects are as follows.

1. The project is the first step to solve this problem and hence concentrates on modelling only primitive shapes like cuboid and cylinders. Most of the objects could be seen as a combination or extension of these primitives.

2. The long term scope which is not achieved in the current iteration is to be able to model a class of objects which can be a combination of these primitives(cylinders,cuboid,cones). Any object which exhibit uniform cross section and are symmetrical about an axis.

3. To be more precise only simple cylinders or cuboid which have a uniform cross section along the axis can be modelled.

4. By being successfully able to model a cuboid/cylinder from a given image and minimal user interaction(4 mouse clicks), one can generate a 3D Model saved as a *.ply* file

5. The model generated is validated by superimposing the model on the image to obtain the same perspective/view with the same dimensions as seen on the screen subject to minimal numerical errors.

6. The image is considered to be an orthographic projection of the model which has to be estimated.

7. The 3D model obtained will not be of the exact dimensions of the object, but will maintain the aspect ratios of the edges as seen in the image.

8. Not all objects can be generated by this process. Objects with flattening edges fail. Objects where the cross sections change from one primitive cannot be modelled. Example: Toothpaste tube

4 Description of the Approach

This section describes the process used to generate 3D Models from a single image of an Object(Cuboid or Cylinder). The crux of the approach lies in the fact that 3D model generation can be done smoothly with user interaction rather than trying to solve it as a Computer Vision problem of fitting a parameterized model to an image[4]. There are other techniques which do a Pose estimation of the object with a standard model[2] [3].

The technique is pretty simple. Given an image of a Cuboid/Cylinder, the user picks *four* points on the screen. For the case of the cuboid, the first point corresponds to origin and the next three points corresponding to the extents in the three dimensions of X,Y and Z. For the cylinder case, the first two points corresponding to the extent of the *Major Axis* and the third point marking the

height and the last point picks one endpoint of the *Minor axis*. The order of picking the points in both the cases are important.

That deals with the input for the system, and that is all is required to genarate a 3D Model. There are two outputs, one is the Model saved as a *ply* file, which is a standard for representing 3D mesh. It contains the Vertices and Faces information. The second output is the rendering of the model on the screen which is superimposed on the image to validate the degree of resemblance.

5 Model Generation

This section deals with the computations involved for the 3D Model creation and the assumptions which are made. The usual approach which people try is to determine the actual Projection Matrix which is responsible for the image, by which they try to estimate the model. This technique forces the user to have a collection of models with which the projections could be computed.

The approach followed here is to compute the actual endpoint of the model in three space by assuming that the model is made of a combination of such primitives. The vase shown in *Figure 1* is an example of an object to be visualised as made up of cylindrical cross-sections of varying diameter. Similarly the cuboid primitive.

Figure 1: An image of a Vase

Key Assumptions

- The first assumption and the most important one is that, the image is an orthographic projection of the model in three space. As orthographic

projection is assumed, there is no scaling of the model based on the relative distance to the viewer.

- The user picks the point on the screen, which is in two space, but the crux of the method described assumes that these points give the actual orientation of the image in 3D. i.e. The User picks the point based on his perception of the orientation of the object in the actual world coordinates.
- This method implicitly assumes there are four *key points* which are visible to be picked on screen by the user.

5.1 Cuboid Model Generation

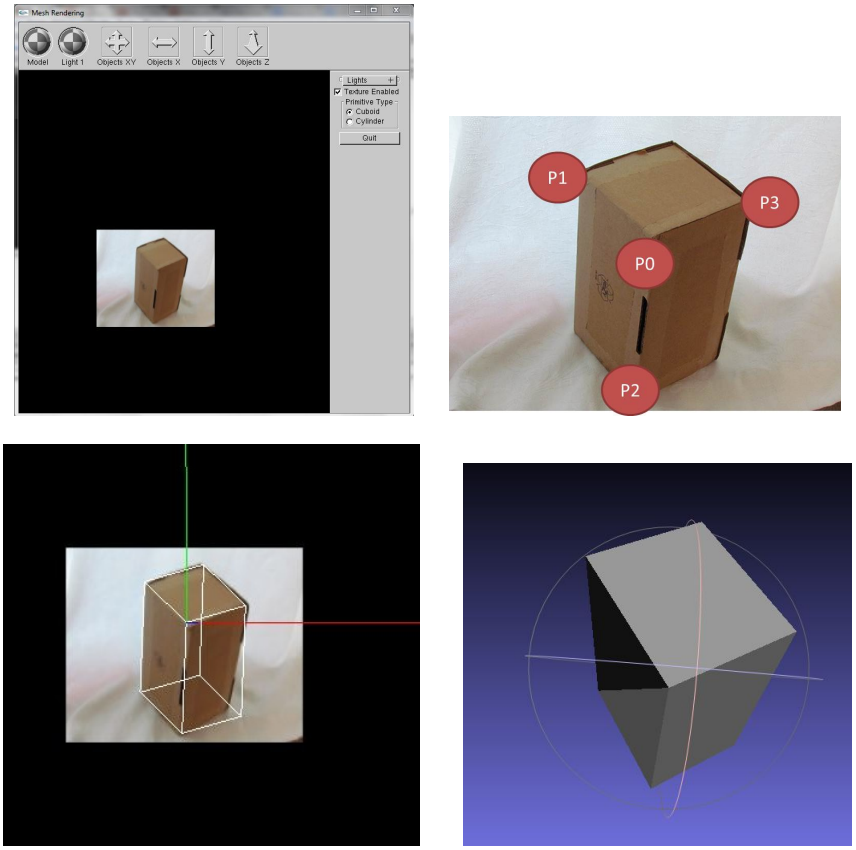

Figure 2: Cuboid Modelling from a single Image

The *Figure 2* describes the process of model generation for the cuboid primitive. The user picks four points as shown in the figure. *P0* corresponds to the origin. *P1,P2,P3* are the endpoints of the cuboid. The order of picking does matter.

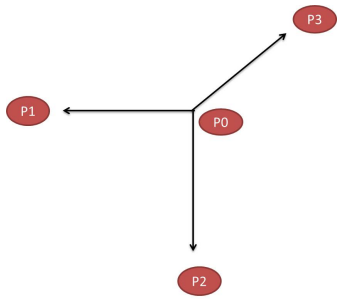

Figure 3: Mutually orthogonal basis vectors

In Model Space the three vectors are mutually perpendicular. Hence the dot products of the three vectors taken pairwise is zero. The points are picked as shown in figure 2. The x,y co-ordinates of these four points are known. One z co-ordinate can be chosen as the origin P_0 in this case. Using the dot product equations, we have three equations and three unknowns which can be solved for.

$$\overrightarrow{P_0P_1} \cdot \overrightarrow{P_0P_2} = 0$$

$$\overrightarrow{P_0P_2} \cdot \overrightarrow{P_0P_3} = 0$$

$$\overrightarrow{P_0P_3} \cdot \overrightarrow{P_0P_1} = 0$$

Figure 2(c) shows the wireframe (white colour) superimposed on the image for validation of the model generated from the vertices computed as described above.

After obtaining the three dimensions of the cuboid, the remaining five vertices are computed as resultants. *Figure 2(d)* shows the visulaizatin of the actual model written into the *ply* file.

5.2 Cylinder Model Generation

The cylinder is generated in the fashion similar to the Cuboid case as discussed above with a few differences. For a cylinder to be computed, one needs to describe the *radius* and the *axis* vector.

In the cylinder case, the user specifies the major axis. It is assumed that the major axis lies in the plane of the projection. So essentially one would see an ellipse which is the projection of a circle in world co-ordinates. As shown in *Figure 4.a* the input image is specified. The user picks the four points. The cylinder wireframe gets computed and superimposed on the image to check for its validation.

P_0 is chosen as the origin and P_1 marks the extent of the major axis as shown in *Figure 5*. From this, the Centre P_c can be computed and the radius

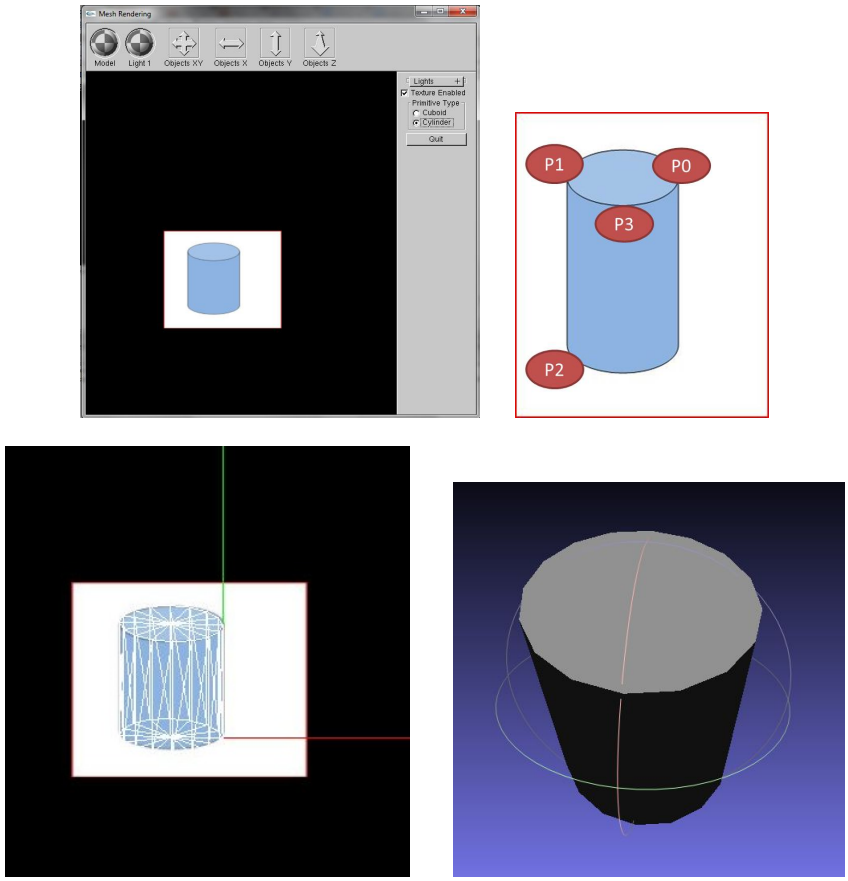

Figure 4: Cylinder Model from a single Image

vector is obtained as $\overrightarrow{P_cP_1}$.

$\overrightarrow{R} = \overrightarrow{P_cP_1}$
$\overrightarrow{Axis} = \overrightarrow{P_2P_1}$

The minor axis we know is the projection of the circle whose radius was $\overrightarrow{R}$. We can compute the Z co-ordinate of P_3 from this relation.

$\parallel \overrightarrow{R} \parallel = \overrightarrow{P_cP_3}$

We compute the circle in the three space by finding unit vectors along the *Radius* vector and $\overrightarrow{P_cP_3}$ using the following equation of the circle.

$$\overrightarrow{C(t)} = \parallel \overrightarrow{R} \parallel * (\hat{u_1}\cos(\theta) + \hat{u_2}\sin(\theta))$$

The next circle is computed by translating the endpoints of this circle along the *axis* vector. The translation is along the direction of $\overrightarrow{P_2P_1}$. The Z co-

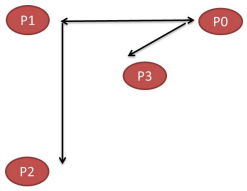

Figure 5: Vectors computed from the picked points for the cylinder

ordinate of P_2 is computed using the fact that

$$\overrightarrow{P_2P_1} \cdot \overrightarrow{P_3P_0} = 0$$

6 Source Code Overview

This section gives a brief overview for important methods in the source code. The code makes use of open-source libraries like *GLUI* for the UI, *SOIL* for loading textures and OpenGL for rendering.

There are two options on the UI, to choose from a primitive type. The respective core methods *computeCuboid() computeCylinder()* gets called appropriately after the first four screen clicks.

Each of these methods generate the vertex list for the *ply* file after the computations described above and call the method *write_file(vlist,primitive)* which takes in the Vertex list and primitive types as the parameter and computes the face list and dumps the result into a *ply* file which is a standard used to store 3D meshes.

7 Future Work and Enhancements

This project is just the first step to the entire 2D to 3D modelling problem. Proving that such a technique works quickly and efficiently with primitives like Cuboid and cylinder allows us to take it further.

1. Only one object is modelled given an image, this could be easily expanded to modelling multiple objects given a scene.

2. The orthographic projections used could be changed to perspective projections by applying a perspective projection matrix to the vertices and faces obtained

3. To increase the class of objects which can be modelled one can use this piece with image processing from libraries like *OpenCV*. Given an image, the edges are detected which mark the boundary of the object(after discarding the noisy edges). They are connected to form contours. To be able to model a vase as shown in *Figure 1*, It could be thought of multiple

cylinders with varying radii. So the extent of the radii can be bounded by the edges detected.

4. The system now makes use of picking screen points, the user interaction could be simplified using a primitive and a drag and drop utility which allows the user to interactively fit the circle to the ellipse in the image.

5. The model generated currently doesn't handle textures. It could approximate a texture from the picked points to render the model to match with the model in the image.

8 Challenges

The biggest challenge was to overcome the highly computationally intensive methods of finding the inverse projection matrix given an image. Also there are very few works which process a single image and try to build 3D models without making assumptions about the model space.

9 Conclusion

This project describes a simple approach to generate 3D Models from a single image along with user interaction. It makes reasonable assumptions to get there and is quick in its computations.

References

[1] Tao Chen. 3-Sweep: Extracting Editable Objects from a Single Photo.

[2] Daniel F DeMenthon. Model based pose estimation in 25 lines of Code.

[3] DANIEL P HUTTENLOCHER. Recognizing Solid Objects by Alignment with an Image.

[4] David G. Lowe. Fitting Parameterized Three-Dimensional Models to Images.