*i*3D: Interactive Planar Reconstruction of Objects and Scenes

Adarsh Kowdle apk64@cornell.edu Cornell University

Yao-Jen Chang yc682@cornell.edu Cornell University

Tsuhan Chen tsuhan@ece.cornell.edu Cornell University

ABSTRACT

3D reconstruction from 2D images is an active research topic in the computer vision community. Classical algorithms like Structure From Motion (SFM) and Multi-View Stereo (MVS) are known to provide a robust framework for reconstruction, but fail to produce a dense plausible reconstruction. Some recent works have shown that making planar approximations of the scene help obtain more complete and visually pleasing results albeit, still approximate. However, many of these algorithms depend on image level cues like strong edges and lines which may be absent in textureless scenes. This has led to interactive 3D reconstruction algorithms.

In this work, we present *i*3D, an interactive planar reconstruction algorithm. In our proposed system, we make planar approximations of the scene and with the help of user interactions, reconstruct the scene given as few as five to ten images from varied poses. As opposed to other more involved interactive algorithms which require the user to provide plane boundaries and line models of the scene, we relax this requirement to mere *scribbles* on an image to indicate the surfaces. Preliminary results show that these simple interactions can help obtain complete and visually pleasing reconstructions, and even handle featureless surfaces in the scene.

Index Terms— Interactive 3D reconstruction, image-based modeling, structure-from-motion

1. INTRODUCTION

3D reconstruction from multiple images is an active research topic in the computer vision community. There has been significant success with automated algorithms [1–6] as well as interactive algorithms [7–10]. Automated algorithms based on structure from motion, recover the camera poses and a sparse point cloud reconstruction of the scene from the image sequence. Dense multi-view algorithms can use the sparse reconstruction to generate a dense mesh model [2, 11]. Some of these works [3, 4] are geared towards video while some [1] are geared towards unordered photo collections on the internet. These results are impressive, but require a large photo-collection.

When the number of input images is restricted, these automatic algorithms fail to produce a dense plausible reconstruction. Multiview stereo algorithms like patch based multiview stereo [2], try to improve the reconstruction by expanding the point features to patches. However, the reconstruction is still incomplete. In order to improve the reconstruction, some of the automated algorithms make planar approximations to the scene [5, 6, 11]. This allows for more visually pleasing reconstructions. However, these algorithms use geometric cues in the images like strong edges and lines which may be absent in textureless surfaces (like walls, etc). This has led to interactive 3D reconstruction algorithms.

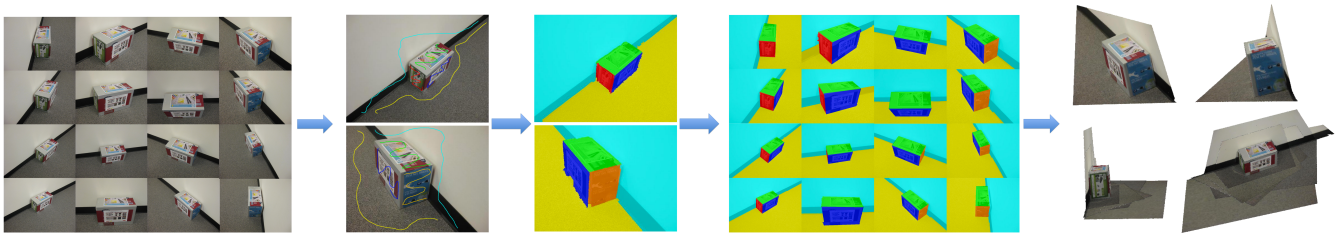
Several interactive reconstruction algorithms have been proposed in literature ranging from providing feature correspondence, to providing plane boundaries and line models of the scene. One such system called VideoTrace [8] is used to interactively model geometry from video. It has a tracing interface and is capable of using information recovered by structure from motion. The user in this system indicates the planes by tracing polygons across the frames of the video. The 2D interactions are converted to 3D reconstructions by using geometric information obtained by applying structure from motion on the video. The user can correct any mistakes in the reconstructions by moving forward in the video and moving the vertices of the polygons. Another interactive 3D reconstruction algorithm was proposed by Sinha *et al*. [9], where in addition to the geometric information from structure-from-motion they use cues like vanishing directions to help reconstruct the scene. The system is similar to VideoTrace, in that the user uses the interface to mark the planes in the scene by providing a line model of the planes. The system in addition allows the user to look at the 3D model in intermediate stages to help correct the line model drawn.

These are clearly very involved interactive algorithms, we relax this requirement to mere *scribbles* on an image to indicate the surfaces. Scribbles have been used in interactive algorithms in the past. They were first used by Boykov *et al*. [12] for interactive segmentation where indicates foreground and background. Srivastava *et al*. [13] improve the 3D model obtained using their Make3D algorithm by using scribbles to enforce coplanarity in their MRF formulation. Sinha *et al*. [9] used scribbles for texture synthesis where scribbles over an occluding object helps remove the occluding object in the visualization by using texture from the other views. The use of scribbles in our system is most related to Boykov *et al*. [12] than the other works. We use scribbles from the user to indicate the planar surfaces and then, use intuitive algorithms to finally create the 3D model of the scene.

We now discuss our approach in detail. We describe the preprocessing steps applied to our collection of few images in Section 2, followed by a description of the interactive algorithm in Section 3. In Section 4 we describe our texture synthesis and the 3D visualizations rendered by our system.

2. PREPROCESSING

In this section, we describe the preprocessing performed on the images before we allow the user to provide interactions.



(a) (b) (c) (d)

Fig. 1: Overview of system: (a) Collection of images given to the system; (b) User providing interactions to indicate the planes in the scene and the resulting segmentations for those images; (d) Resulting plane cosegmentation after all the refinement; (e) Some sample novel views of the model produced (Best viewed in color).

Structure From Motion (SFM). In our SFM algorithm, we detect features in all the images and match features across all the image pairs. We use two of the images for relative pose estimation, to estimate the pose of these two cameras and estimate the positions of these matched features in the 3D point cloud. We then perform incremental structure from motion [1] where we estimate the position of other points in the 3D point cloud and the camera poses for the other views in an incremental fashion. We use intrinsically calibrated cameras, so, the camera poses are estimated by minimizing the error between the reprojected 3D points and the 2D feature points using the perspective-n-point (PnP) approach. The locations of the triangulated points in the 3D point cloud and optimally *adjusted* using bundle adjustment [14]. At the end of this step we have, the camera projection matrices for all the views, the reconstructed 3D point cloud and list of the points visible by each camera.

Superpixel map. We use mean shift [15] to break down the images into about thousand superpixels per image.

Daisy descriptors. In [16], Tola *et al*. showed 'SIFT-like' daisy descriptors which serve as dense local image descriptors can be used to obtain a dense depth-map in wide baseline stereo applications. In our algorithm, we use the daisy descriptors computed for every pixel to help characterize the content of every superpixel by averaging the descriptors over all the enclosed pixels. This is later used to classify the superpixels into the planes in the scene.

3. ALGORITHM

In this section, we first describe the interface we have built to accept the user interactions. We then describe how we use the interactions from the user to achieve the segmentation of the image into the planes followed by how we finally used this information to create the 3D model of the scene.

3.1. Interface and interactions

We have created a java based user interface for our system. The interface has been developed so it can work on any touchscreen based machine in addition to the conventional machines. The source code for this GUI may be downloaded from [17]. The system gives the user a set of images and allows the user to select any image. Once selected, the user proceeds to scribble on the image with different colors indicating different plane surfaces in the scene as shown in Figure 2. These scribbles are used by our algorithm to segment the image into the different plane surfaces as explained in the next section.

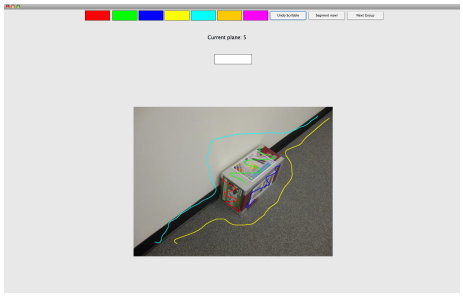


Fig. 2: The user interface to provide scribbles to indicate the plane surfaces (Best viewed in color).

3.2. Scribbles to segmentation

We start with the image the user scribbles on, and use these scribbles to build a multiclass classifier capable of classifying every superpixel in the image into one of p planes labeled. We use all the superpixels which have been scribbled on by the user. The descriptor for each of the labeled superpixels is computed as the mean daisy descriptor of all the pixels in the superpixel.

We train a one-against-all classifier using a logistic regression model. We now test every superpixel in the image by using the regression model for each plane label. In order to maintain some smoothness, we formulate the plane labelling problem as energy minimization defined over a graph constructed over the image. We build a graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, over superpixels, with edges connecting adjacent superpixels. The energy over this graph can be defined as follows:

$$E = \sum_{i \in \mathcal{V}} E(X_i) + \lambda \sum_{(i,j) \in \mathcal{E}} E(X_i, X_j), \tag{1}$$

where the first term is the *data term* indicating the cost of assigning a superpixel to a particular plane, while the second term is the *edge term* used for penalizing label disagreement between neighbours. The negative log response of the logistic regression is used to model the data term for each superpixel. We set the dataterm of the superpixels scribbled by the user to $-\infty$ (in practice a large negative value) as hard constraints in the energy minimization. We model the edge term using the Potts model, a smoothness term commonly used in energy minimization methods [18–20],

$$E(X_i, X_j) = \mathrm{I}\left(X_i \neq X_j\right) * \beta, \tag{2}$$

where $\mathrm{I}(\cdot)$ is an indicator function that is 1 if the input argument is true (and zero otherwise), β is a scale parameter.

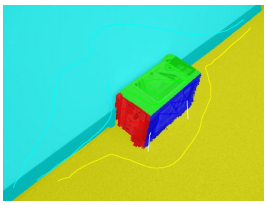
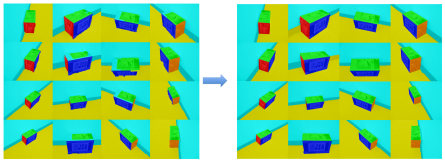


Fig. 3: Sample edge connectedness scribbles, used to achieve globally optimal plane parameters by enforcing that the plane pair share these edge pixels (Best viewed in color).



(a) (b)

Fig. 4: Segmentation refinement: (a) Original segmentation of the planes with superpixel leaks (black circle); (b) Refined segmentations (Best viewed in color).

3.3. Plane co-segmentation

The minimum energy labelling obtained using graph cut, segments the image into the different planes labeled by the user as shown in Figure 1b. We use the 3D point cloud and their 2D point correspondences obtained using SFM, to transfer the labels from the 2D images to the 3D points. We now use RANSAC-based plane-fitting [21] on the labeled 3D points to estimate the plane parameters of the labeled planes. At this stage we use additional interactions from the user to indicate the edges shared by two planes by easily scribbling two lines across the edge as shown in Figure 3. We use these as constraints while estimating the plane parameters by enforcing that these boundary points lie on both the connecting planes, thus, resulting in globally optimal plane parameters. These contraints help estimate plane parameters even for textureless surfaces like the wall in Figure 1.

Using the plane parameters and the camera projection matrices estimated during the preprocessing stage, we can define a homography from the image which was scribbled on, to every other image. We use these homographies to warp the plane segments in the scribbled image, and hence transfer labels onto the other images. The segmentations of the group of images are as shown in Figure 4.

3.4. Closing the loop

Our system allows the user to select an image and provide the scribbles to indicate the plane surfaces. Having estimated the planes to approximate the scene, the system can display the plane segmentations and the 3D model. The user can now provide more scribbles to improve the model and indicate new, previously occluded planes, thus closing the loop on interactive 3D reconstruction.

4. TEXTURE SYNTHESIS

In this section, we discuss two forms of 3D visualizations our system produces. First, we describe a dense color mapped point cloud obtained by back-projecting all the pixels onto the 3D surfaces. Next, we describe a mesh based visualization which involves creating a mesh by triangulating the scene in 3D and overlaying the texture on the mesh. The two methods we discuss have tradeoffs and depending on the intended use of the model the user can select one or the other.

4.1. Refining segmentations

An important preprocessing to the texture synthesis in our system is refining segmentations. A well known problem with using superpixels in vision applications is superpixel 'leaking' similar to the leak shown with a black circle in Figure 4a. We handle this problem with a two stage segmentation refinement.

First, using the homorgraphies we warp every segment in every image to every other image. Now, for every image, we take a pixelwise vote across the warped segments for the plane labels and suitably modify the plane segmentation. We observed that our visualization was affected by the fact that we relied on the pixelwise refinement to define the segments which resulted in jagged boundaries. In order to refine this, we consider the lines in 3D corresponding to the intersection of every pair of planes. These 3D lines are projected onto the images, and the segmentations are refined using these line boundaries resulting in sharper plane boundaries. The refined segmentations are as seen in Figure 4b. This second stage of refinement however, would work best with planar surfaces with polygonal boundaries.

4.2. Back-projection algorithm

Given the plane parameters of the surfaces in 3D, we seek to determine the position of every 2D image point on the 3D plane. We develop the back-projection algorithm using the camera geometry, by evaluating the point of intersection of the ray from the camera center through the pixel on the image plane, and the estimated 3D plane surface [22]. This formulation is the basis for the visualizations we describe.

4.3. Texture mapping

At the end of the plane co-segmentation stage, we have the plane segmentations for every image. For each image, we use the plane segmentation, and for every plane surface in that image compute the 3D positions of the corresponding pixel locations using the back-projection algorithm described above. It is clear that, using texture from one image would result in artifacts like holes due to the perspective view of some of the images, on the other hand, using texture from all the images can result in visually annoying changes in illumination across images and inter-meshing of textures.

We avoid these artifacts by first normalizing the luminance channel of all the images and then performing view selection for each planar surface. We rank the views for each plane based on angle between the plane normal and viewing direction *i.e.* lower this angle, better the view. Using the texture from the best view for every plane surface can give a good visualization however, it would not cover the occluded regions of the plane. So, for each plane surface we start with the best view and back-project the pixels corresponding to that plane. We then move to the next best view and back-project the pixels from the non-overlapping regions of the plane. This is continued till all pixels are accounted for, or we run out of views.

4.4. Visualizations

4.4.1. Dense point cloud visualization

The dense point cloud based visualization is similar to [1], however with the difference that, we have the 3D model expressed as planes which can be used to map texture from all the image points and not just the feature points. We use the texture mapping algorithm described above and map the texture from the images onto the

(a) (b) (c)

Fig. 5: Mesh visualization: (a) Dense point cloud visualization; (b) Single mesh model; (c) Multiple mesh model (Best viewed in color).

corresponding voxels. We use an OpenGL based viewer which can render the dense point cloud given the 3D voxel locations and their corresponding texture from the images. This visualization is shown in Figure 5a.

4.4.2. Mesh visualization

The mesh visualization is similar to [23, 24]. This visualization requires triangulating the 3D points corresponding to the boundaries of the planes and overlaying the texture from their projections on the image. We generate a VRML file which can use the mesh and the corresponding image textures and render the texture mapped mesh model. We made some interesting observations with the mesh visualizations. Many of the earlier works in 3D visualizations like [5, 23, 24], create a single mesh from an image, which results in the 3D model shown in Figure 5b. This single mesh model can make the visualization very pleasing to look at, hiding artifacts with perspective views discussed before, however, this does not cover regions of the scene occluded in that image. This would require representing the model as a combination of multiple meshes *i.e.* a mesh for every plane resulting in a model as shown in Figure 5c. This would however leave artifacts on the wall when compared to the dense point cloud visualization.

We can thus conclude that, if we need to create a complete static model, we would have to represent the model as a dense point cloud, however, if we would like to create fly-throughs in the scene [5, 6, 23, 24], we can render the texture as a single mesh from one image. Some additional results on building datasets are shown in Figure 6.

5. CONCLUSIONS

In this paper, we propose a new interactive planar 3D reconstruction algorithm which uses simple user interactions in the form of *scribbles* to indicate the planar surfaces in the scene. We discuss multiple geometry based methods to refine segmentations, and even fix superpixel leaks in the process. We also describe two methods of visualizations used. Preliminary results show that these simple interactions can help obtain complete and pleasing reconstructions and, even handle featureless surfaces in the scene.

6. REFERENCES

[1] N. Snavely, S. Seitz, and R Szeliski, "Photo tourism: Exploring photo collections in 3d," in *SIGGRAPH Conference Proceedings*, New York, NY, USA, 2006, pp. 835–846, ACM Press.

[2] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multi-view stereopsis," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2009.

[3] M. Pollefeys, D. Nistr, J. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. Kim, P. Merrell, C. Salmi, S Sinha, B. Talton, L. Wang, Q. Yang, H. Stewnius, R. Yang, G. Welch, and H. Towles, "Detailed real-time urban 3d reconstruction from video.," *International Journal of Computer Vision*, vol. 78, no. 2-3, pp. 143–167, 2008.

(a) (b)

Fig. 6: Results on building datasets: Novel views after a single mesh 3D model has been rendered (Best viewed in color).

[4] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, "Visual modeling with a hand-held camera," *International Journal of Computer Vision*, vol. V59, no. 3, pp. 207–232, 2004.

[5] S. Sinha, D. Steedly, and R. Szeliski, "Piecewise planar stereo for image-based rendering," in *ICCV*, 2009.

[6] Y. Furukawa, B. Curless, S. Seitz, and R. Szeliski, "Reconstructing building interiors from images," in *ICCV*, 2009.

[7] P. Debevec, C. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach," in *SIGGRAPH*, 1996, pp. 11–20.

[8] A. Hengel, A. R. Dick, T. ThormŁhlen, B. Ward, and P. H. S. Torr, "Videotrace: rapid interactive scene modelling from video.," *ACM Trans. Graph.*, vol. 26, no. 3, pp. 86, 2007.

[9] S. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys, "Interactive 3d architectural modeling from unordered photo collections," *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2008)*, 2008.

[10] "Google sketchup: http://sketchup.google.com/," 2000.

[11] Y. Furukawa, B. Curless, S. Seitz, and R. Szeliski, "Manhattan-world stereo," in *CVPR*, 2009.

[12] Y.Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images," *ICCV*, 2001.

[13] S. Srivastava, A. Saxena, C. Theobalt, S Thrun, and A. Y. Ng, "i23 - rapid interactive 3d reconstruction from a single image," in *Vision, Modelling and Visualization*, 2009.

[14] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *Vision Algorithms: Theory and Practice, LNCS*. 2000, pp. 298–375, Springer Verlag.

[15] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *PAMI*, vol. 24, no. 5, pp. 603–619, 2002.

[16] E. Tola, V. Lepetit, and P. Fua, "Daisy: An efficient dense descriptor applied to wide baseline stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 99, no. 1, 2009.

[17] D. Batra, A. Kowdle, K. Tang, and T. Chen, "iScribble, `http://chenlab.ece.cornell.edu/projects/iScribble/iScribble.html`," 2009.

[18] D. Batra, A. Kowdle, K. Tang, D. Parikh, and T. Chen, "Interactive cosegmentation by touch, `http://chenlab.ece.cornell.edu/projects/touch-coseg/`," 2009.

[19] J. Cui, Q. Yang, F. Wen, Q. Wu, C. Zhang, L. Gool, and X. Tang, "Transductive object cutout," in *CVPR*, 2008.

[20] A. Criminisi, T. Sharp, and A. Blake, "Geos: Geodesic image segmentation," in *ECCV*, 2008.

[21] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, June 1981.

[22] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2000.

[23] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 824–840, 2009.

[24] D. Hoiem, A. Efros, and M. Hebert, "Automatic photo pop-up," in *ACM SIGGRAPH*, August 2005.