3D Image Reconstruction from Stereo Images and Single Images

Todd Macdonald
Stanford University
tmacd@cs.stanford.edu

Wilbur Yang
Stanford University
wilbury@cs.stanford.edu

Abstract

We attempt to reconstruct 3D scenes from stereo images by selecting 3D models from a dataset and placing them in the scene. Specifically, we segment the input stereo images into silhouettes of objects and search for the closest 3D model in our database using Zernike descriptors. Then, we orientate the 3D model properly and place it in the scene. Whereas previous work has depended on depth information from active depth sensors or 3D scanners, we attempt to reconstruct scenes using only stereo images. With our system, we can ideally reconstruct semantically plausible scenes from images, enabling rapid generation of 3D scenes for applications such as virtual reality. We currently work with 10^4 *models in our 3D model database.*

Although we do not have many results yet, the ones that we do have are promising and often semantically make sense. As an extension, we may apply the same ideas to 3D reconstruction from a single image if our 3D model database is labeled such that each model is associated with its real world size.

1. Introduction

The reconstruction of 3D scenes can be described as taking as input a pair of 2D stereo photos and producing as output a scene of 3D points constructed using those photos.

In our approach to this problem, instead of merely solving for the 3D points of the features in the images, we select and fit 3D models to the contours obtained from segmentation of the input images.

This approach is useful in practice since 3D reconstructions that depict a scene using only reconstructed points without the additional use of pre-made 3D models are incomplete and greatly prone to error. Compared to pure 3D construction, 3D reconstruction with 3D model retrieval allows for one to create immersive environments without a plethora of extremely accurate sensor data.

This type of model-based reconstruction is used extensively in industry to recreate outdoor scenes. For instance, Google has used pre-made 3D models of buildings to populate its maps on Google Earth. In the academic community, several papers such as Funkhouser *et al.* have pioneered using 3D model retrieval to turn 2D sketches into 3D models.

2. Previous Work

2.1. Review of Previous Work

Many attempts to solve similar problems have already been made in this well-researched topic.Perhaps most similarly to our study, Chen, Lai, *et al*. perform 3D scene reconstruction from RGB-D input images obtained using active depth sensors (Kinect) [2].

Furthermore, using contextual information, they add constraints to the objects selected by 3D model retrieval such that the generated output scenes are semantically reasonable. Because we are working with inputs of higher error (stereo images as opposed to active 3D scans), we expect for the results from this project to be a target for our study.

The two major components to our project have been explored separately in other papers:

1. That of retrieving the correct 3D models from 2D images.

2. That of reconstructing the 3D scene using the found 3D models.

Regarding the first component, several 3D model retrieval systems have been created in the past. Chen, Tian, *et al*. describes the process of building a 3D model retrieval system based on Funkhouser *et al*.'s initial approach. Whereas in the earlier paper spherical harmonic descriptors were used to identify similarities between 3D models and their contours, Chen, Tian, *et al*. instead explores the use of Zernike moment descriptors and Fourier descriptors [4] [1].

These descriptors are found to work well due to their rotation and scale invariance. They have both implemented systems which allow users to draw a 2D silhouette to retrieve any 3D mesh. Both papers achieve speeds of under one second for retrieval from a 3D model dataset containing on the order of 10^5 models, employing optimized computation of the respective descriptors' coefficient similarities.

For the second component, the state of the art locates and orientates 3D models by leveraging scanner data. Kim *et al.* uses 3D scans of indoor scenes to perform faithful reconstructions of large, cluttered indoor environments [6]. It does so by first fitting planar patches to the 3D point cloud, then using RANSAC to fit primitive shapes such as boxes and cylinders to those patches, and finally learning to group sets of primitives into objects.

Fisher *et al*. uses scene understanding to generate plausible 3D scenes [3]. By segmenting the 3D point cloud into planes and attempting to understand the purpose of each plane (i.e. for holding up objects vs. for holding up humans), they are able to semantically classify parts of objects (e.g. as "gaze," "touch," "back support," and "hip support"). Using these categorizations, the relevant 3D models are retrieved and added to the scene.

Because we are working with 2D stereo images, it is difficult to incorporate all of these 3D point cloud techniques into our study. However, it is useful to keep in mind the extent to which 3D point cloud data can benefit a 3D scene reconstruction.

2.2. Novelty of Our Study

In our study, we experiment with methods involving 3D model retrieval from images alone. Our work is novel in that the depth data is computed from the stereo images. Related work, such as that from Chen, Lai, *et al.*, performs 3D scene reconstruction from RGB-D images. Unlike previous papers, which have used 3D scanner or depth sensor information in their reconstructions, we use only 2D stereo images to perform the reconstruction, while still recognizing the orientation and depth of the objects in the 2D images. In other words, by comparison, in our study, the input images have only RGB values. We compensate for depth by by calculating the disparity map of a pair of stereo images, from which we can derive depth values.

In addition, our proposal to work with single images and a labeled 3D model dataset has not been attempted to the best of our knowledge.

3. Methods

3.1. Summary of Technical Solution

We base our procedures off of those in Chen, Tian, *et al.*, as well as Funkhouser *et al.*

When recreating our 3D scene from a pair of stereo images, we first precompute several pieces of information. To begin with, we take our database of 3D models and then form orthographic projections of each of the models. We currently use 6 orthographic projections per model, which consists of the six sides of the object.

Once the projections have been calculated, we calculate the Zernike moment descriptor and Fourier descriptor of each projection.

Once this precomputation has been completed, we then take an input of a pair of stereo images. Using the OpenCV library that utilizes the block matching algorithm, we form a single disparity map from the two input images. From this disparity map, we are able to segment the different objects that are present in the original pictures and create contours from the segments. This segmentation is performed by binning different areas of the image by their respective depths in the depth map.

Next, once our images have been segmented into individual objects, we calculate the Zernike and Fourier descriptors for each object. By computing the Euclidean distance between the descriptor for our 2D contour and the descriptors for each of our 3D models, we are able to identify which 3D model matches our object from the 2D contour best.

Once we have located our corresponding 3D model for the object, we then place this object in our reconstructed scene. The x and y coordinates are the same as in our input images. To calculate the depth, we use the depth map calculated from before. Each 3D model in our database has its orientation saved, so we are also able to reproduce the orientation of the object.

For the more ambitious approach, when using only a single image, we are able to find the depth at which to place a model by measurement information that is provided in our labeled dataset. By knowing the actual height and width of our models, we can attempt to approximate the depth at which they would exist in a 3D scene.

3.2. Overview of Technical Solution

3.2.1 Datasets

We work with the NTU 3D Model Database, which consists of 10^5 3D models scraped from the Internet. As the models originate in the "wild," there are many different categories of objects, ranging from airplanes to fish to flowers.

This dataset originates from Chen, Tian, *et al*.

The Middlebury Stereo Dataset consists of about 40 rectified stereo images, which we use as our inputs.

3.2.2 Frameworks and Architectures

Our architecture consists of C++ code. We use openFrameworks (specifically ofxAssimpModelLoader to load, process, and render 3D models), Eigen (for orthographic projections and image transformations using matrices), and OpenCV (used classes including StereoBM to create disparity map, used Mat class to represent and manipulate images (read image, change color scheme, create new image with additional channels of information)).

3.2.3 Scene Segmentation Framework

3.2.3.1 CIELab Conversion

Based off of the reasoning in Dal Mutto *et al.*, we propose convert our RGB input images into the CIELab color space. Instead of measuring the amounts of red, blue, and green light as with RGB, CIELab measures the color lightness, as well as the location on the spectrum between magenta and green and yellow and blue.

For segmentation, CIELab offers several benefits over RGB. Most importantly, CIELab is a more perceptually uniform color space, which means that changes to the values of the 3-dimensional color vector are more proportional to the visual changes that a human would perceive [**?**]. This attribute of the CIELab color space enables us to better segment an image, as color vectors corresponding to each pixel will be a greater euclidean distance apart for visual differences. In our color transformation, we also normalize our color vectors by the standard deviation of the color lightness.

3.2.3.2 Stereo Reconstruction

To better match the 3D models to the objects in our 2D stereo input, we derive depth information from the pair of input stereo images that describe our scene. These depth values enable our segmentation algorithm to be more accurate, as the segmentation algorithm is given an additional channel of information (ie, depth) with which to perform the segmentation.

We calculate these depth values using a disparity map. A disparity map is a one channeled image whose pixel values indicate to the level of disparity between locations in the left and right stereo images. Pixels with greater disparities are closer to the camera; thus the disparity map can be used to gauge depth in a scene from a pair of stereo images.

For our implementation, we use the StereoBM class in OpenCV. This class utilizes a version of the block matching algorithm to produce a disparity map. The block matching algorithm itself is fairly straightforward.

The block matching algorithm is as follows. Consider the window as divided into many windows. These windows are also sometimes referred to as blocks. We then create a new image of the same size as the input images and then populate it with the disparity values corresponding to each window in the left stereo image.

Figure 1. Reconstructed disparity map of a piano from the 2014 Middlebury Stereo Image Dataset.

These disparity values are calculated by finding which windows in right image most closely correspond to the windows in the left image. When finding the correspondence between two windows, you start by examining the same window locations in both images. Next, you sum the square distance between the pixel values in one window with all windows in the other image that are within a certain distance. The window that has the smallest sum of square distances is now considered the best corresponding window.

This process is repeated for many iterations, with the most closely corresponding window in the previous iteration becoming the center of the search in the next iteration. The farther apart distance-wise the two corresponding windows are, the darker that window of pixels will be in the disparity map. Therefore, the color of the pixels in a disparity map is a proxy for depth.

3.2.3.3 Segmentation via Meanshift

The segmentation of our 2D stereo images is critical, as we need to be able to isolate the individual objects in our scene in order to run our matching algorithm and retrieve the 3D models with which to place in our scene. To perform this segmentation, we use the meanshift algorithm with six different channels of information: x, y, and z position coordinates, as well as l, a, and b color values (see CIELab color conversion above). All of these values, save for the z coordinate, are taken directly from the left stereo image. To obtain the z coordinate corresponding to each pixel in the left stereo image, we use the depth values from the disparity map that we calculated previously.

However, due to time constraints, we segment the images manually by looking at depth values, resulting in segments that resemble that of meanshift for images involving flatter objects.

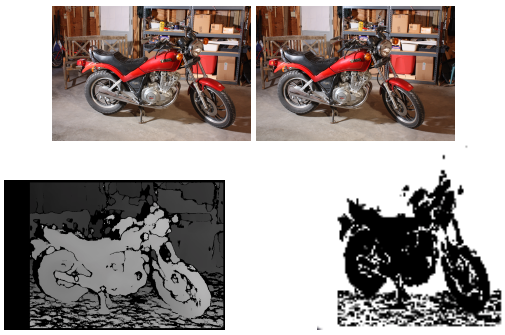

Figure 2. A stereo image of a motorcycle, its stereo reconstruction, and a segment of the motorcycle.

3.2.4 Orthographic Projections of 3D Models

In our matching algorithm, discussed below, we are matching a pair of 2D stereo images with the closest corresponding 3D model. To accomplish this matching, we in effect match a pair of 2D stereo images with a 2D orthographic projection of a 3D model.

To perform the 2D orthographic projections of our 3D models, we center, we first read in a file on a 3D model. We convert the mesh triangles describing the 3D model into a series of 3D points. With these points, we then translate the points so the model's center is at the origin and then appropriately scale the model. Finally, we use a rotation matrix to transform each point in the model into a variety of orientations, from which we take an orthographic projection. The figure below shows some example orientations of the same 3D model.

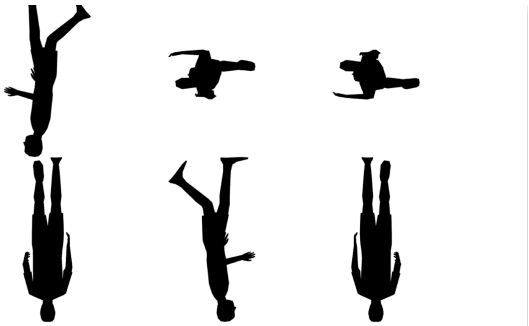

Figure 3. Several orthographic projections of varying orientation of a man walking from the NTU 3D Model Database v1.

Since the objects in our scene may contain objects that are in non-standard orientations, we take 6 orientations of each 3D model for use in our matching algorithm. By taking several projections of the same 3D model, we help ensure our matching algorithm is robust enough to handle for different orientations.

3.2.5 Matching Algorithm

Our matching algorithm is fairly straightforward. First, we calculate the 36-dimensional Zernike moment descriptor for a segment from our pair of stereo images. This segment describes an object for which we are trying to find a corresponding 3D model. Next, calculate the distance between the Zernike descriptor of our segment and the pre-processed Zernike descriptors of all of the orthographic projections of our models from the MLU 3D Model dataset. The model whose descriptor is the closest euclidean distance from the descriptor of the segment is considered a match.

We chose to use Zernike moment descriptors for several reasons. The descriptors are invariant to scale and rotation, are robust to minor errors, and efficiently represented, as each descriptor entry is derived from a polynomial that is orthogonal on the unit disk.

4. Experiments

4.1. Performance of Matching 3D Models with 3D Models

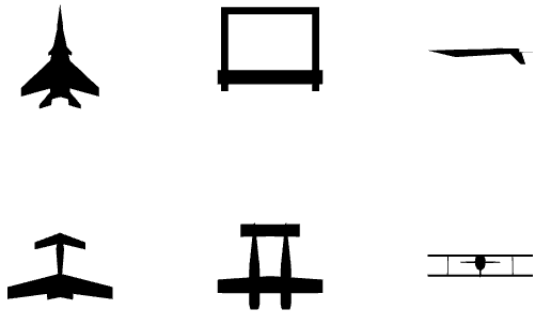

Figure 4. Some 3D models (top) matched with the closest 3D models (bottom).

The matching of 3D models to 3D models seems to make sense visually, even with a low number of 3D models used.

4.2. Performance Matching 3D Models with Stereo Images

We see potential in the matching of segments to 3D models. The instance below is a match which makes semantic sense.

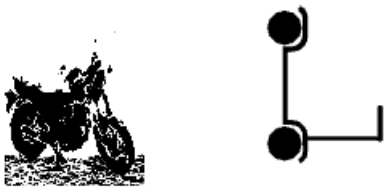

Figure 5. A motorcycle segment and its 3D match (a scooter, rotated 90 degrees).

In other cases, the match makes less sense, such as an umbrella to a plane or a recycle bin to a plane. Since the segments are noisy, and the subset of 3D models that we used includes many plane models, this result is undesirable but makes sense.

4.3. Performance Tradeoffs Between Using Stereo Images and Single Images For Scene Reconstruction

We have not yet implemented the single image reconstruction, but we forsee the following challenges:

In terms of matching to the correct model, the disparity map enables the mean shift algorithm to run with one additional channel of information (depth). Without this information, our segments will have more error than in the stereo version.

In terms of placement in the scene, the single images require a labeled dataset, which is hard to obtain for a dataset of size 10^5 models downloaded from the Internet. In addition, the single images require accurate image segmentation to effectively calculate depth (if the image segmentation has dimensions that do not represent the object in the image, then the calculated depth will be wrong)

5. Future work

More work could be done in ensuring that the 3D model match is semantically correct. Occlusion and stereo reconstruction / segmentation error presents challenges in identifying the correct models to place in the scene. Perhaps implementing a semantic graph like in previous works will benefit this model greatly.

The code for this project can be found at `https://github.com/wilburyang/cs231a-project`.

References

[1] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung. On visual similarity based 3d model retrieval. In *Computer graphics forum*, volume 22, pages 223–232. Wiley Online Library, 2003.

[2] K. Chen, Y. Lai, Y.-X. Wu, R. R. Martin, and S.-M. Hu. Automatic semantic modeling of indoor scenes from low-quality rgb-d data using contextual information. *ACM Transactions on Graphics*, 33(6), 2014.

[3] M. Fisher, M. Savva, Y. Li, P. Hanrahan, and M. Nießner. Activity-centric scene synthesis for functional 3d scene modeling. *ACM Transactions on Graphics (TOG)*, 34(6):179, 2015.

[4] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs. A search engine for 3d models. *ACM Transactions on Graphics (TOG)*, 22(1):83–105, 2003.

[5] W.-Y. Kim and Y.-S. Kim. A region-based shape descriptor using zernike moments. *Signal Processing: Image Communication*, 16(1):95–102, 2000.

[6] Y. M. Kim, N. J. Mitra, D.-M. Yan, and L. Guibas. Acquiring 3d indoor environments with variability and repetition. *ACM Transactions on Graphics (TOG)*, 31(6):138, 2012.

[7] D. Zhang, G. Lu, et al. A comparative study of fourier descriptors for shape representation and retrieval. Citeseer.