Image Based Renderer

Timothy Liu
University of California at Berkeley
timothytliu@gmail.com
EE290T – 18945232

ABSTRACT

Image Based Rendering (IBR) is an open research field with many techniques on recreating a 3D world representation. The techniques can be classified with respect to the amount of input images and how much is known about the scene geometry. One common technique is to use the plenoptic function that models a 3D dynamic environment by recording the light rays at every space location towards every possible direction. This requires very little information on the known geometry of the scene, but an enormous amount of input images. Another common technique that relies heavily on explicit geometry is used in more traditional computer graphics rendering. This requires texture maps, depth-maps and a classical rendering pipeline to account for all of the inputs for representing the 3D environment. This paper takes a hybrid approach by tagging each image with a pose describing the viewing angle and normal vector of the camera. A simple, scalable, real-time, and interactive rendering system is presented in this paper.

General Terms

Perspective Projection, Homography, Correspondences

Keywords

SIFT Features, Image Based Renderer, RANSAC

1. INTRODUCTION

The Image Based Renderer as described in this paper has the following functional specifications. A 3D environment would be composed of input images, correctly transformed to account for the movement of the viewer. This includes translational and rotational movement of the viewer within the scene. Furthermore, a real-time constraint required that images would need to be displayed dynamically instead of rendered on initialization. An additional task would be to increase the field of view of the viewer by leveraging corresponding features found in images. Finally, the images would have tagged pose information that would allow for a better rendering as the user moves throughout the scene. The renderer has three primary components: an image model, the rendering model, and the movement model.

2. IMAGE MODEL

2.1 Image Data

The data set consisted of 784 images at 1345 by 1007 pixel resolution. These images spanned an entire hallway enumerating an entire world. As the renderer would need to display an image corresponding to the user's point of view, a single image is often not enough. Therefore, in addition to using each image to map to a specific point of view, images can be stitched together or mosaic-ing to provide an increased field of view.

2.2 Feature Detection

Although there are various ways to combine two images, a common technique is to find features that are present in both images. A homography is needed to transform one image into the perspective of another image. Using the common features and locations present within each image, the point correspondences help to determine the transformation matrix. The homography is characterized by a 3x3 matrix with 7 degrees of freedom. Although only four point correspondences are needed to generate the homography, it is often impossible to account for degenerate features or redundancies in the image. If some point correspondences lie on the same line, then no solution may arise from the linear system of equations. SIFT features were used to characterize each image with a neighborhood value of.

2.3 Homographies between Features

After finding a sufficient number of features for every image in the test set, the next task was to generate homographies between each pair of images. The algorithms section describes the technique used to estimate the best homography. The homography can also be characterized by how close an image is to one another. This helps to give a metric on how to choose the transformations when rendering. The distance metric used in this paper is a simple norm calculation of the difference between the homography and the identity transformation. Therefore, at this step in the process all the images have been sufficiently characterized by features and can be correctly rendered in the perspective of another image.

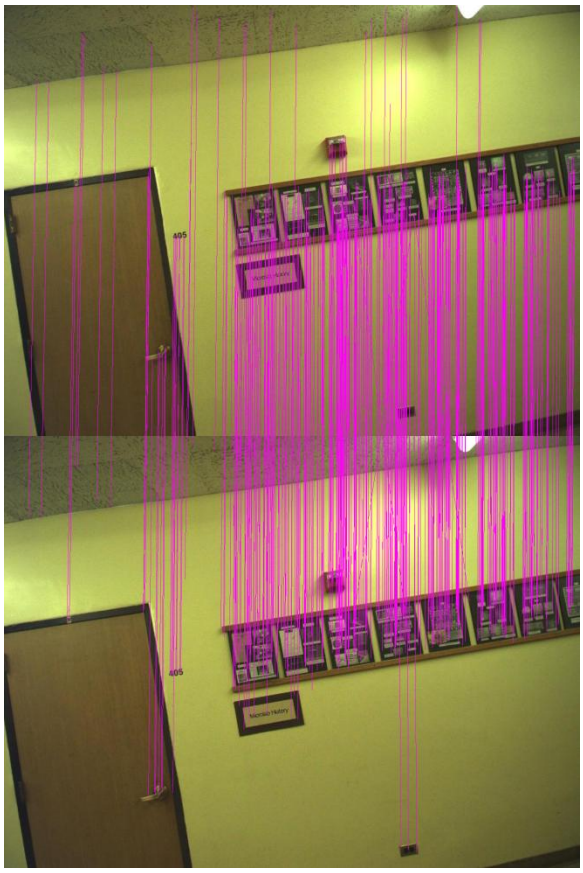
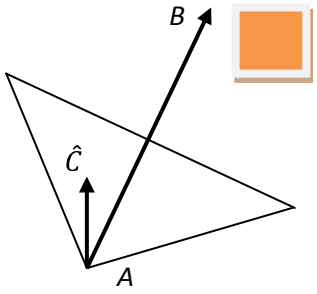


Figure 1. SIFT Features and their Point Correspondences

Figure 2. Mosaic formed using homographies



A: Camera Origin, B: Center of Projection, C: Up or Normal Vector

3. RENDERING MODEL

3.1 Projective Geometry

There are many rendering schemes and the task to render these images is unique. The main problem associated with rendering two-dimensional images in three dimensions is that there is no depth. The user can only see a projection of the scene, which also must reflect how the user is oriented. Therefore, projective geometry is used quite extensively in figuring out what image to render and how to orient the image correctly. The data set also includes associated pose information for each image. The pose describes the camera axis along which the scene is projected onto the image plane.

Furthermore, an up vector or the vector normal to the camera axis determines the orientation of the camera as it can still rotate along the camera axis. Every image has a corresponding pose that describes the image. A scene pose describes the user within the image based renderer because the user also has a specific orientation in how to view the environment. The pose is the underlying structure used in all the perspective geometry.

3.2 Rendering Pipeline

The rendering pipeline is specifically geared towards rendering images in real-time. Currently, the system runs at 10 frames per seconds, but can achieve up to 20 frames per second. The first task is to determine which image to render or more specifically, which image best corresponds to the current pose of the user. The following step is to transform the image into world space from object space. The image if viewed independently is always 1345 by 1007 pixels and perpendicular lines in the image may be skewed.

However, in world space, the image is transformed to render the object regardless of where the camera was that took the picture. This serves as an intermediate step to calculate the perspective transformations that align the image to the user's pose and not the world space coordinate frame. Finally, the actual rendering of images occur with an additional feature.

Along with the nearest image, neighboring images are also rendered in the coordinate frame of the viewer. Thus, the viewer has an increased field of view and can see more than a single image. This step utilizes the previously calculated homographies to determine how to render neighboring images. Each step in the rendering process takes significant time for computation; therefore, simple and elegant algorithms were used to perform each task.

3.3 Algorithms

closestImageIndex:

1. Order spatially local images
   - $score = (c_i - e_i) \cdot (c_j - e_j)^{-1}(c_i - c_j)$
2. Search within local images
   - $argmin\ score(viewer, j)$

modelViewMatrix:

1. e = eye, c = center, u = up
2. Find rotation around z-axis
   - $f = (c - e) - (c - e) \cdot u\hat{u}$
   - f = forward vector projection
   - fAngle = atan(f[1], f[0])
3. Find rotation around x and y axis
   - upAngleX = atan(up[1], up[2])
   - upAngleY = atan(up[0], up[2])
4. Return product of matrices

SIFT Features + RANSAC function:

1. Find features for image *i and j*
2. Search all combinations of features for correspondence
   - $\sum_{k=1}^{length}(f_{i1}[k] - f_{j1}[k])^2 < THRESHOLD$
   - satisfies => correspondence
3. Given set of N correspondence points, estimate H with RANSAC
   - Select a random sample of 4 correspondences and compute H
   - Calculate each distance, $d_\perp$
   - Calculate # of inliers for H where $d_\perp < t = \sqrt{5.99}\sigma$ pixels
4. Pick H with largest # of inliers

4. MOVEMENT MODEL

4.1 Translation

The user has the option to move in the up, down, left, and right directions. However, the rendering of the images needs to reflect the changes in the viewer's pose accurately. Therefore, the mapping of the translational movement is defined as:

$$(\Delta y, \Delta z) = (\Delta c_y, \Delta c_z)$$

$$H_{viewer} = \begin{vmatrix} 1 & 0 & \Delta y \\ 0 & 1 & \Delta z \\ 0 & 0 & 1 \end{vmatrix}$$

4.2 Rotation

Rotation is mapped differently because the pose is kept in rectangular coordinates while a rotation is an angle. Therefore, the following relationship is drawn between the angle of rotation and the position of the viewer's eye and center points.

$$\Delta\theta \rightarrow (c_x, c_y) = (cos(\Delta\theta), sin(\Delta\theta))$$

$$H_{viewer} = \begin{vmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) & 0 \\ \sin(\Delta\theta) & \cos(\Delta\theta) & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

4.3 Zoom

Zoom can be described as moving forward along in the x direction; however, it is also described as scale of the overall image. The zoom is mapped within the rotational components to maintain all information within the viewer's pose. The map can be described as such:

$$zoom \rightarrow (zcos(\theta), zsin(\theta))$$

$$zoom = \sqrt{(zcos(\theta))^2 + (zsin(\theta))^2}$$

$$H_{viewer} = \begin{vmatrix} z & 0 & 0 \\ 0 & z & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

5. PROJECT RESULTS

5.1 Field of View

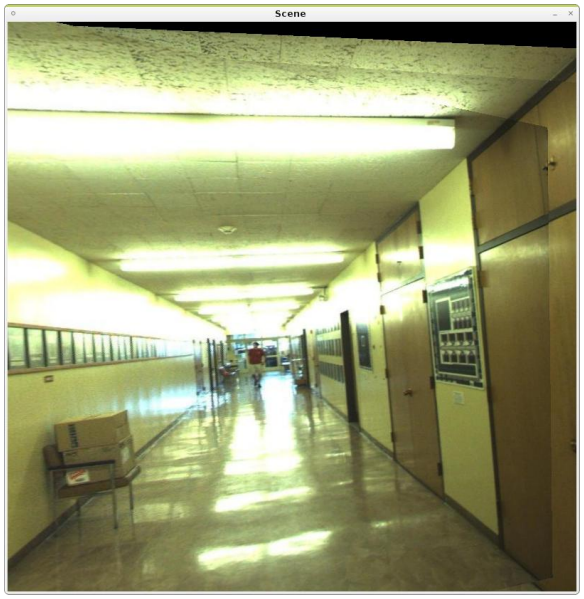


Figure 3. Dynamic Rendering of Increased View

5.2 Demo and Statistics

The result of this experiment was a demo that although not perfect, demonstrated the core functionalities. The demo rendered 263 images from the data set and functioned at an average rate of 10 fps. The movement model contained zooming, translational, and rotation movement around the z-axis although the rotation was limited to 90 degrees. The demo was not fully capable of moving within the entire three-dimensional range of motion; however, the user could travel within the entire space of the images. The viewer experienced an increased field of view for every frame that was dynamically rendered upon movement. The mosaic was sufficiently detailed to allow for a smooth stitching of images. Finally, the renderer functioned fairly simply among it's components.

263 Images	Render Time
Initialization	0.0
Loading Data	18.0
Generate Features	17.0
Generate Matches	216.0
Generate Norms	0.0
Search Homography	0.0
Rendering	Variable
Total – Rendering:	251.0

Table 1 – Statistics for Final Demo

6. CHALLENGES

I have experienced various challenges from deciding on a platform/language (Linux + C) to including necessary packages (OpenCV, SIFT, OpenGL). However, the biggest challenge I have faced so far has been the large data set and real-time constraint.

6.1 Scalability

The sample data set has approximately 1000 images, which may turn into an arbitrary large number in the future. Currently, I store all the images that the renderer requires in memory, and I am running 24 parallel threads and cache features to disk. However, I am still running out of memory and can only run approximately 300 images in my system. Therefore, to have a truly scalable system, I would need to look into dynamically loading and storing images upon render. A potential solution would be to remove degenerate or redundant images where the minimal set of images serves as input to the final environment.

6.2 Real-Time Rendering

Another significant concern is the real-time user interaction. The current system is not fully functional without full three-dimensional range of movement and a truly optimized spatially local search. However, each of these components needs to be run every frame. Therefore, there is a bottleneck for the ideal frame rate. Assuming 15 frames per second, that leaves approximately 60 milliseconds to perform all the computation of searching for the best image, transforming the image to the viewer's perspective, mosaic-ing the images, and the final render of a current 800 by 800 pixel wide window. Therefore, creating a real-time system is a significant challenge.

7. CONCLUSION

The renderer was successful in accomplishing its functional objectives, albeit not completely. Specific issues such as scalability and real-time interaction remain unsolved although approximated. The transformations perform correctly, except there is not full functionality for the entire range of motion. However, what is left can be extended easily from the current transformation and rendering process.

Future work would focus first on eliminating small sources of error with the current test set. The next step would involve optimizing current design to make the system fully scalable. Finally, achieving an ideal 30 frames per second would be the last goal in creating a fully functional interactive three dimensional image based renderer. There is much work left to be done in terms of image based renderers in general, and it was extremely beneficial to create this current system.

8. ACKNOWLEDGEMENTS

I would like to thank Professor Avideh Zakhor and Matthew Carlberg for their continued assistance on this project. Both their helpful guidance and knowledge helped me

9. REFERENCES

[1] Hartley, Richard and Zisserman, Andrew. Multiple View Geometry in Computer Vision. Cambridge University Press. 2003.

[2] Wikipedia. Wikimedia Foundation, Inc. http://en.wikipedia.org/