Light Field Photography for CGI/Live Action Video Integration

Matt Lathrop
Stanford University
Department of Computer Science
mlathrop@stanford.edu

Keenan Molner
Stanford University
Department of Electrical Engineering
kmolner@stanford.edu

Abstract

To separate out objects in a live action video sequence, an artist must step through the footage, frame by frame, and rotoscope out the occluding object from the background where a 3D computer generated graphic is to be inserted. This slow and expensive process can be remedied with the use of light field photography to capture depth information with every frame of film. These depth maps can be applied as a texture offset to the film in a 3D rendering program and virtual objects can be inserted into the scene at any particular depth, naturally occluding the background and being occluded by objects in the foreground, without any rotoscoping or manual editing. In this paper, we investigated two different methods of lightfield photography to replace rotoscoping.

1. Introduction

With incredible progress in computer graphics and 3D rendering, many films undergo some form of editing to alter the live action footage, from replacing the background, to recoloring the light, to creating alternative worlds, or even giving fantastic superpowers to the characters. Typically, the use of a green screen allows digital artists to subtract the green background out of a film frame and insert a new background into the frame. Additionally, artists can easily overlay textures or graphics in front of the live action. But what happens when a virtual object needs to occlude some figures in the scene and be occluded by others? How do live action characters interact with virtual characters? Since most films are only 2D projections of 3D scenes, it's left up to computer artists to step through each frame of the video and create the depth maps that allow for the live action and virtual occlusions, interactions, and dynamics.

1.1. Rotoscoping

Rotoscoping refers to the frame-by-frame identification of objects in a scene. Once an artist has rotoscoped some live action footage, virtual objects are able to be placed into the scene and the artist can choose which objects should and should not occlude the virtual object.

While the process sounds easy, it is actually a very time consuming and difficult process. During the process an artist goes frame-by-frame in a video and manually masks by hand each object that will need to occlude a virtual object. While there are tools such as tracking, edge detection, and more to help artists, it is still a major expense to any film which requires actors to interact with virtual objects.

After talking with several people within the film industry we determined that the process of rotoscoping was one that few would miss. The elimination of that process could provide as big of a jump in the quality of films as was achieved by the first computer generated animations over hand drawn animations.

1.2. Lightfields

A lightfield is a function that describes each point in the 2D projection of the scene as a ray with a spatial direction, rather than just a single pixel in a 2D plane. By shifting the perspective of the rays that describe the scene, the focus of the image is retunable in post production. Additionally, by examining how one object in the scene changes its perspective relative to the different viewpoints, the distance of that object from other objects in the scene can be calculated.

Lightfields can be constructed with any combination of 2D images that looks at the same scene from multiple viewpoints. This can either be a result of a 2D sensor with an array of lenslets to map all rays from one perspective to a specific region of the sensor, with other rays from other perspectives mapped to other parts of the sensor. With each frame captured, the depth of each frame can be reconstructed from multiple viewpoints on the sensor. Typically, images shot on a camera with a lenslet array are of lower spatial resolution since one silicon sensor is used to capture all the different perspectives of the scene. This method is used in consumer lightfield cameras like the Lytro camera.

Alternatively, a 2D camera without a lenslet array can photograph the same scene from multiple angles along the

X and Y directions by sliding the camera along a track in that dimension. A lightfield can be constructed by combining these different perspectives on the same scene. Unlike the lenslet approach, each perspective on the scene is captured individually, rather than all at once. If a character or object in the live action film only appears in one perspective of the scene, they will not be accounted for in the light field, as this method looks to compare images of the same scene from multiple perspectives. This method does typically offer images with higher spatial resolutions since the only one perspective is captured on the sensor at a single instance, rather than the entire lightfield multiplexed across the sensor.

2. Related Work

There have been numerous research efforts along the path of combining live action video with virtual objects. Since a highly detailed and accurate depth map is required to use in 3D compositing, many research papers have looked at high resolution depth map construction.

A group from Disney Research [2] has published a paper on calculating high resolution depth maps from multiple perspectives on the same scene. Their method looks at the edges of features of the scene extruded through cross sectional slices of the light field image stack and calculating the slope of the features as they change in perspective. The more change a pattern demonstrates across the bird's eye view of the lightfield stack, the greater distance the feature is from objects at the center of the stack. Both an object that lies very close and very far from the center show great change in distance across the lightfield stack, but in opposite directions. By calculating the slope of these lines, the relative depth of one object from another can be extrapolated. Additionally, Disneys method starts by examining areas with highest spatial frequency and then explores larger, more homogeneous regions in a fine-to-coarse manner, rather than many other image processing techniques that evaluate from coarse-to-fine, smoothing along the way and reducing the resolvability of fine details.

The Fraunhofer Institute [3] constructed a high spatial resolution camera array to capture lightfields in their paper Multi-camera system for depth based visual effects and compositing. They used an array of cameras along with a main camera that used a beam splitter to allow both cameras to see the scene from the same angle. While the images captured provided nice high resolution images, their rig was large and their depth estimation based on a limited number a viewpoints.

The current standard for consumer lighfield cameras is the Lytro Camera, which uses a single sensor and a lenslet array to multiplex the perspectives across the sensor [4]. This camera runs a proprietary depth calculation algorithm that allows for depth map creation and image refocusing of each frame. A camera like this, but with a higher pixel density and faster readout times could be used to film a scene and add in composited effects, in post. Unfortunately, in practice, we found the calculated depth maps from the Lytro camera to be underwhelming and often make gross oversimplifications of occlusions and regions with high spatial frequencies. While the hardware is compact, the image resolution is low, since only a single sensor is used and the algorithm for depth calculation is too sloppy than whats required for film compositing.

3. Project Overview

For our project, we implemented the fine to coarse depth map algorithm described in the Disney Research paper in MATLAB. With this algorithm, we attempted to reconstruct depth maps from the data set that accompanied Disney's paper. We then imported these depth maps into a 3D compositing software and mapped the depth map to the texture displacement map of the the video frame and inserted a 3D virtual object into the scene. For comparison, we also constructed a camera track of our own and shot an image sequence with a Lytro camera. Like before, we composited the depth map that came out of the Lytro camera onto the video footage and inserted a 3D object into the scene.

3.1. Depth Algorithm Implementation

The depth algorithm developed by Disney [2] begins with still images of the same scene taken from multiple angles along the x-axis. The image sequence is then stacked together in the z-direction and slices are cut out in the y-direction to form epipolar images from the x-z plane, showing the change of velocity of a particular object by the slope of the line it forms in the epipolar plane. The algorithm compares the radiance of neighboring pixels to find a slope line that best approximates the edges of the epipolar image. This slope can be turned into a depth in the z-direction from the relationship $m = \frac{z}{fb}$ where m is the slope of the line in the epipolar plane, z is the distance form the object to the center of the image stack, f is the focal length of the camera, and b is the disparity between two images in the sequence. This process is illustrated in Figure 1

The algorithm calculates all the depths it can make a strong estimation for in the first iteration, then downsamples the image in the x and y directions and begins the process again. As we downsample from iteration to iteration, all regions of low contrast begin to look like edges across the epipolar slices and can be approximated with the depth algorithm above. Once depth values are assigned, the downsampled image is upsampled and these depth values are propagated along all images in the scene. They are then un-flagged and will not be recalculated in subsequent downsampling and estimations.

Figure 1. Disney Process

For each iteration, we calculate all points in the epipolar plane above a specific threshold that could be edges. This edge confidence matrix for each epipolar slice is represented by

$$C_e(x,z) = \sum_{x' \in N(x,z)} \|E(x,z) - E(x',z)\|^2$$

where $N(x,z)$ is a small, 1 dimensional window in the plane E around a point (x,z).

We then try to calculate a depth for all edges in this confidence matrix by varying a hypothetical disparity (slope) across all points along the center of the epipolar plane and calculate a score for how well this hypothetical depth fits with the image by summing radiance values along this line. Disparity guesses with higher radiance values are more likely to represent constant edges in the epipolar plane. Radiance values along a line are the average of the Red, Green, and Blue channels at a pixel represented by the equation

$$R(x,d) = \{E(x + (\hat{z} - z)d, z) | z = 1, \ldots, n\}$$

where n is the number of images in stack and d is the hypothetical disparity. We calculate a depth score $S(x,d)$ for each estimation with the equation

$$S(x,d) = \frac{1}{|R(x,d)|} \sum_{r \in R(x,d)} K(r - \hat{r})$$

where r is the radiance at that specific point in $R(x,d)$ and $K(x)$ is the kernel $1 - \|x/h\|^2$ if $\|x/h\| \leq 1$ and 0 otherwise. h is a threshold for us to vary the confidence - we used an h of .02 for our algorithm implementation. The depth we assign to the pixel is the value with the highest score in $S(x,d)$ based on the depth estimate:

$$D(x,\hat{z}) = \arg\max_d S(x,d)$$

Our confidence in this depth estimation is calculated by

$$C_d(x,\hat{z}) = C_e(x,\hat{z}) \|S_{max} - \bar{S}\|$$

where S_{max} is the $\max_d S(x,d)$ and $\bar{S} = \sum_d S(x,d)$.

To eliminate outliers, we replace our depth estimate $D(x,\hat{z})$ with the median value of the set formed by a small window across neighboring epipolar slices in the y-direction. Once we have filled in a estimated depth for a specific pixel in the x and y directions across the z-stack, we un-flag these pixels, smooth our image, downsample, and rerun our algorithm. In this next iteraiton, we will only be making estimates for the flagged pixels without depths, which will increase our computational speed.

When this iteration returns a value, we upsample this image back the original image size and fill in all missing pixel information that this iteration could provide. We continue this downsample-calculate-upsample process until we've downsampled to an image with less than 10px in a given direction. At this point, we fill in all missing values with the depths calculated at this lowest sample size.

Finally, we apply a 3x3 median filter to all x-y images to remove any speckle. At this point, we have now approximated a depth value for every pixel in the x-y plane across all images in the z-direction.

3.2. Image Capture

We constructed a 1m long linear track and mounted the Lytro camera to the track. With this linear track, we shot a 100 frame film of stationary objects moving across a table. With some objects near to the camera and others further from the camera, we observed the occlusion of the far objects by the near object. The objects had both low and high spatial frequencies, used in the accuracy comparison of the depth maps from the Lytro camera algorithm vs. the Disney algorithm.

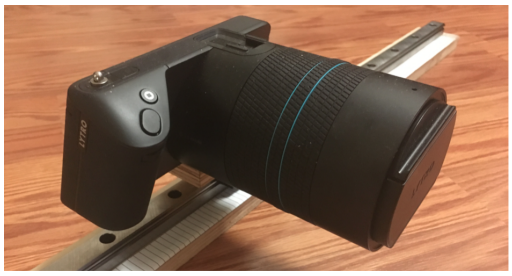

Figure 2. Camera Track with Lytro Camera

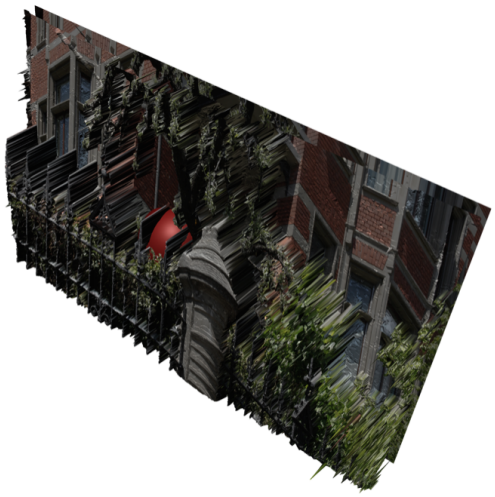

Figure 3. A perspective view of the depth map applied to the 2D image sequence

3.3. Composition

The Autodesk Matchmove software package was used to analyze the video sequences filmed with the live action camera and recreate a virtual camera that followed the same motion path as the camera used to film the scene. This virtual camera was then focused on a 2D rectangular plane that played back the live action footage. The virtual camera and motion path were then exported to Autodesk Maya for 3D compositing. The depth maps calculated by both the Disney algorithm and by the Lytro algorithm were imported and the depth value at each pixel of the image sequence was used as the displacement texture applied to the rectangular plane showing the live action film. The live action frame was converted from a single rectangle to a spatial array of polygons with the displacement texture generated by the depth map [1].

Maya currently lacks the ability to render polygon textures based on changing displacement textures from frame to frame, which unfortunately resulted in the need for manual pipelining and re-rendering for each frame of the film. Methods to improve the implementation of the depth displacements are discussed in later sections.

Next, we created a spherical object in Maya and colored it and sized it appropriately for the scene. By looking at the top view of the displacement map, we moved the sphere along the Z depth axis until it was in the middle of the scene, occluding background objects and occluded by the foreground.

With the virtual camera set to film the textured and polygon version of the frame, rather than the 2D rectangular projection, we rendered out the image sequences from the Disney and Lytro datasets. We did not move or change the depth of the sphere object from frame to frame, but rather left the occlusions and depth displacements as a result of the depth maps applied to the live action video sequence.

4. Results

We were able to render a 3D composition for both depth calculation methods. Our MATLAB implementation of the Disney algorithm did not work as well as the original algorithm created at Disney, but did pick up the most important features of the images. With more time and some consultation with the original researchers, we are confident we could recreate their image processing pipeline.

4.1. Disney Algorithm

The original algorithm was performed with 101 4080x2720 pixel RGB images and computed on the GPU of a NVidia GTX 680 graphics card. The entire image stack took Disney 9 minutes. Our non-optimized algorithm running on the CPU took 8 hours and was run with a downsampled image stack in the x and y directions to 2040x1360. While we found that higher resolution images did produce better depth maps, we were computationally limited to downsampled images.

While our algorithm successfully constructed depth maps for areas with lots of detail and high spatial frequency, we were not able to accurately calculate the depths for large homogeneous regions. While the algorithm is supposed to downsample the image until all homogeneous regions look like edges, calculate a depth, and fill in any missing depth values from previous iterations, we were not able to propagate our estimated depths from the final iteration all the way back to the full resolution image.

We rendered our 3D composition using the depth maps provided in the Disney dataset. The results are show in Figure 4.

Figure 4. Disney implementation v. our implementation

4.2. Lytro Algorithm

The Lytro camera provided a depth map for each frame of the video sequence. However, since no data from one frame is propagated to surrounding frames, the distances from object to object were all relative. As a result, the depth value from the camera to a single point in the scene would change from frame to frame. We corrected for this by implementing a script that normalized out all the values to a specific point in the live action sequence, such that all depths were now relative to this point.

5. Discussion

Each mode of capturing a depth image and using this information to replace rotoscopting offers its unique advantages and disadvantages. While we were able to construct 3D scenes using both depth maps, neither imaging modality provides a perfect depth map to use in all cases.

Since the implementation of the Disney Algorithm calculates the distances from the camera to the scene by comparing the edges seen in the same scene from multiple perspectives, the algorithm works best for static scenes. The depth map for the frame uses the data from all other frames captured. A stronger coherent signal from frame to frame, results in a depth map of more confidence and higher detail. We found 2D images with spatial resolutions allowed for better, more detailed depth maps than down sampled versions of the same scene. Additionally, the more images in the sequence, the more perspectives are incorporated into the lightfield of the same scene. When we varied the number of images we incorporated in our depth algorithm, we found that more images resulted in more detailed depth maps.

While this algorithm stands out for capturing the high frequency, nuanced scenic details, like the branches of a tree or the ironwork on the fence, shown in Figure 4, the algorithm does not work well if the details of the scene change from frame to frame, as they would if we were filming a person interacting with the world. An object that appears only briefly in the image sequence is largely disregarded by the depth calculation algorithm, since the edges of this object will only be present in a few frames of the sequence. The algorithm is looking for consistent edges across all sequences and the object's edges will be ignored. This type of algorithm is best when constructing a 3D composition in a mostly static scene.

The image sequence in Figure 5 shows the final composition with a 3D object inserted into the live action image stack. Note the high levels of detail in the occlusions of the sphere. A short image sequence shown as a film can be seen at this website: `http://makeagif.com/i/WC8tLT`.

The Lytro camera captures a depth map for each frame of the video by multiplexing the different perspectives across the same sensor. Each perspective on its own has lower spatial dimensions than any of the frames taken by the 2D camera and processed with the Disney algorithm. Since the depth map algorithm of the Lytro camera is proprietary, we were not able to compare the depthmap of a down sampled image to the depthmap from a standard image.

It is very obvious that the the small details and high frequency content captured in the RGB image is lost in the depth maps. We believe this is a result of both spatially smaller 2D perspectives to examine for edge coherence and fewer perspectives of the same scene to compare across. However, as a tradeoff, the Lytro camera provides a depth map for frames of the sequence with details changing from frame to frame. Unlike the Disney algorithm, objects that change from to frame are not ignored.

While this algorithm stands out for capturing moving objects across multiple frames, the algorithm does not work in areas of high detail.

The image sequence in Figure 6 shows the final composition with a 3D object inserted into the live action image stack taken with the Lytro camera. Note the lack of detail in regions of sharp contrast. A short image sequence shown as a film can be seen at this website: `http://makeagif.com/i/ANGyF_`.

6. Future Work

In this paper, we've demonstrated the potential for lightfield photography to make a large shift to the way the film industry edits films for 3D rendering by effectively eliminating the need for rotoscoping. However, there is still a great deal of work to be done in both the software and hardware used to capture the image sequences.

The long image tracked camera with higher resolution perspectives offered the opportunity for high spatial reconstruction. The Lytro camera allowed for frame by frame depth maps, but with poor depth map quality and low resolution. In order for hardware to capture data to calculate an accurate depth map for each frame, the size of the cameras must increase to accommodate for both of these imaging modalities within the same camera. A larger camera sensor with a lenslet array would offer a wider set of perspectives from which to calculate the depth map off of. Additionally, a larger sensor would allow for higher resolution sensor arrays under each of the lenslets. We found the depth algo-

Figure 5. Disney stack, post compositing

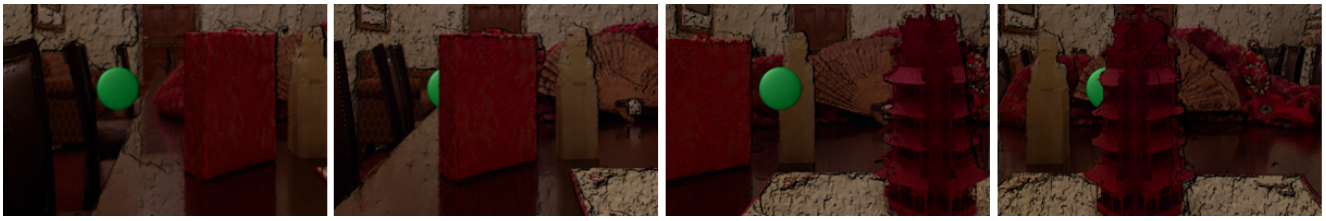

Figure 6. Lytro stack, post compositing

rithms to work better at higher spatial resolutions and with more perspectives on the same scene, both of which would be offered by a larger camera.

When scaling this imaging system larger, more and more data is created for each individual frame. The readout speed of the camera must be increased to maintain a frame rate constraints and larger and faster data storage must be available to meet the growing demand for such data intense image capture.

Software to construct depth maps must also be improved. The Disney algorithm took 9 minutes to run on a GPU. As the size of the image sensors increase, the time it takes to calculate the depth increases exponentially.

The software used for 3D compositing must also provide better support for depth maps, or plugins must be developed. To construct the frames of the videos in this paper, we were required to calculate a polygon mesh with the depth map and apply the RGB video to the polygon mesh. This process of converting a depth map to polygons was not easily automatable, nor was it available for the types of planes showing the image sequence that the virtual cameras are focused upon. Better support for temporally shifting depth maps and their rendering in polygon form must be available in the 3D compositing software before this technique becomes commonplace.

This technique shows a great deal of promise and we look forward to seeing the evolution of hardware and software towards the goal of lightfield photography for cinema and virtual/live action video integration.

7. Special Thanks

We thank Prof. Gordon Wetzstein and Orly Liba for their support with this project. Additionally, we thank Jon Karafin from Lytro and Changil Kim from Dinsey Reaserch.

References

[1] J. Grant. Control displacement maps in maya - displacement mapping.

[2] C. Kim, H. Zimmer, Y. Pritch, A. Sorkine-Hornung, and M. H. Gross. Scene reconstruction from high spatio-angular resolution light fields. *ACM Trans. Graph.*, 32(4):73–1, 2013.

[3] M. Ziegler, A. Engelhardt, S. Müller, J. Keinert, F. Zilly, S. Foessel, and K. Schmid. Multi-camera system for depth based visual effects and compositing. In *Proceedings of the 12th European Conference on Visual Media Production*, page 3. ACM, 2015.

[4] F. Zilly, M. Schöberl, P. Schäfer, M. Ziegler, J. Keinert, and S. Foessel. Light-field acquisition system allowing camera viewpoint and depth of field compositing in post-production. 2013.