Seeing through Obstructions: Obstruction-Free Light-Field Photography

Wanling Liu
liuwl@stanford.edu
Department of Electrical Engineering
Stanford University

Ye Li
liye5@stanford.edu
Department of Electrical Engineering
Stanford University

Kefan Wang
kefanw@stanford.edu
Department of Electrical Engineering
Stanford University

Abstract

We present a new image processing pipeline that reconstructs obstruction-free image from light field images with obstructions. The pipeline involves image data pre-processing, pixel location calculation algorithm, pixel extraction, and pixel filling. Our approach successfully reconstructs the obstruction-free images efficiently and it is robust to various occlusion circumstances. We show results on uniformed color foreground, depth-variant foreground with complex shadowing and continuously changing foreground.

Introduction

It is a common problem of unwanted obstructions when we capture images through a fence, and it can be hard to recover the background scene without attenuation caused by the foreground fence. For example, surveillance cameras would encounter such issue if it is ill-placed behind some obstructions, and the resulted scene may lose its valuable information. Another common situation is that when we take photos in zoos or courts, we want to get rid of the fence to get a clear and unobstructed scene. Fortunately, with a light field camera, such grid-patterned fence can be digitally removed. This is because light field cameras can capture sub-aperture image array where images have different points of view. It enables us to locate the obstruction based on the depth map and then reconstruct the occluded part using information from all the images captured with a single shot.

In this project, we aim to remove the foreground obstruction and recover the background scene as perfect as possible using Lytro images. We implement masking light field method, pixel extraction, and region filling algorithm. After analyze pros and cons of each algorithm including runtime, use cases, output image quality, we propose a novel obstruction-free image processing pipeline that combines two main algorithms together and can be applied in various cases with different occlusion circumstances: uniformed color foreground, depth-variant foreground with complex shadowing and continuously changing foreground, etc. We also fine-tune parameters in our pipeline to achieve better final visual reconstruction results.

Related Work

Removing obstructions has been actively investigated in different aspects with various algorithms. In the papers [Gu+09] and [FS03], researchers discuss how to remove occluders when we have images captured by different apertures. In [McC14], the author raised an algorithm that can optimize the fence mask through propagation. The basic idea of this approach is to find the foreground mask by propagation through all sub-aperture images. It searches for a shift value to obtain a least RMS difference in the masked pixels. Then the clean background can be rendered using this mask. In [Liu+08], authors designed an algorithm to remove the fence-like occlusions using a single image. Their model has three-step: 1) finding lattice (implement the iterative algorithm described in [Hay+06]); 2) separating foreground and background (construct masks using k-means to separate foreground and background); 3) filling the background with textures (implement texture filling method proposed in [CPT04]). Our proposed pipeline was inspired by this three-step model.

There are also works related with videos / image sequences. For example, In [MLY13], they researched on the removal of fence-like occlusions in videos, based on the fact that those fence-occluded pixels tend to appear later in the temporal dimension. In [Xue+15], researchers proposed a computational approach to remove reflecting or occluding elements from a short image sequence captured with camera slightly moved and recover the desired background scene. They proposed a two-step algorithm for obstruction-free photography: initialization and iterative optimization. As for initialization, edge pixels are extracted from the input images to calculate the motion vectors and each pixel is assigned to either the obstruction layer or the background layer. With interpolation, they could obtain a preliminary estimation of motion fields for background and obstruction layer. Then in optimization stage, motion fields, background and obstruction components are updated iteratively until convergence.

Generally speaking, there are plenty of researches on obstruction-free image under different circumstances, but there are only a few investigations on occluder-free reconstruction for light field cameras.

Dataset

We used light field images from Stanford Lytro Light Field Archive: http://lightfields.stanford.edu/occlusions.html. We also captured our own

light field images using Lytro camera by ourselves to obtain images with different occlusion pattern. The captured images have the depth map generated by Lytro camera automatically. We used Lytro Desktop to convert captured images into Lytro Camera Raw Image file, and decoded it using LytroPowerToolBeta which outputs light field images in 16-bit png format in linear color space. This tool also helps compensate the exposure and adjust coordinates with lens aberration and variation of sensor angular sensitivity taken into consideration.

Approaches

The key idea is that the image array and depth map encode different geometry information so that we can use them to reconstruct the obstruction-free image. This idea works because the image array contains images of different views of the scene, and different images are obstructed differently. At a high level, our approach is to pre-process the image, calculate shifts and refine the image.

Pipeline Overview

We propose an obstruction-free image processing pipeline. The input is the raw images and depth map captured by Lytro. We firstly pre-process the data with Matlab and Lytro softwares (*Lytro Desktop, LytroPowerToolBeta*), and then calculate pixel shifts to recover most parts of the image as our intermediate results. After that, we extract the "unfilled pixels" (parts behind the obstruction where the real scene information is not well captured by the light field camera or shifting calculation) and fill those pixels using a pixel filling algorithm. Our final output is a more desirable obstruction-free image.

Masking Light Field Method (MLFM)[McC14]

The basic idea of Masking light field method is to determine the shift of the background and foreground image, noted as δ_b, δ_f, by using the SIFT feature detection algorithm, and the depth map shift match, respectively. For simplicity, let's denote sub-aperture image array as $L^{(u,v)}$, and let $L^{(0,0)}$ be the center sub-aperture image. Similarly, let $M^{(u,v)}$ be the binary foreground mask for corresponding sub-aperture image.

$$M^{(0,0)}(x) = 1(D(x) < threshold) \tag{1}$$

For detecting the background shifts, we can use the well-built SIFT feature detection package(in our case, we used *vlfeat* in Matlab). While getting one unit vertical shift and horizontal shift, use the linearity of shifting among all sub-aperture images to obtain all of the shifts $\delta_b^{(u,v)}$ accordingly.

While the SIFT feature detection algorithm is already coming with a well-built package, the depth map shift for detecting the foreground shifting $\delta_f^{(u,v)}$ can only be done by searching for the least mean squared error between the shifted mask and each sub-aperture image in a window of around 10 pixels, i.e.

$$\delta_f^{(0,1)} = argmin_{\bar{\delta}} \Sigma_x (L^{(0,0,)}(x) M^{(0,0,)} - L^{(0,0,)}(x+\bar{\delta}) M(0,1)(x+\bar{\delta}))^2 \tag{2}$$

Note that in the original paper[McC14], the author assumes the obstruction was in uniform color, so that it was easier to be extracted without the depth map(as shown in Figure 2). But in natural images, there are more cases that the fence itself includes some complex lighting and shading, and we also want to take that case into account. Fortunately, while taking the light-field image, it will automatically generate a rough depth map without further effort, and that's the reason we want to take advantage of the depth map in our algorithm.

Figure 2: Example image of different types of obstructions that need to be considered when constructing the algorithm.

To compute all the shifted foreground mask, we can apply the shift $\delta_f^{(u,v)}$ directly to the center mask $M^{(0,0)}$

$$M^{(u,v)}(x) = M^{(0,0)}(x + \delta_f^{(u,v)}) \tag{3}$$

After these parameters were found, we can simply subtract each sub-aperture image with corresponding shifted mask values, and summing all of the sub-aperture images and take the average to get the correct image.

$$\bar{I}(x) = w(x) * \Sigma_{u,v}(1 - M^{(u,v)}(x+\delta_b^{(u,v)})) L^{(u,v)}(x+\delta_b^{(u,v)}) \tag{4}$$

$$w(x) = \frac{1}{\Sigma_{u,v}(1 - M^{(u,v)}(x + \delta_b^{(u,v)}))} \tag{5}$$

Although this method is intuitively make sense, there are also some problems that need to be taken into account. As shown in Figure 3, with the naive masking light field method, there are some "unfilled pixels" that will not have any values at all. These pixels are the parts behind the obstruction where the real scene information is not well captured by the light field camera or shifting calculation. To resolve this, we proposed to use another algorithm to somehow "predict" the unfilled regions that looks as natural as the known regions.

Pixel Filling Algorithm (PFA)

In order to address the problem of unfilled pixels, we add a pixel filling step after MLFM. The entire pixel filling algorithm (PFA) contains two parts: 1. extracting the unfilled

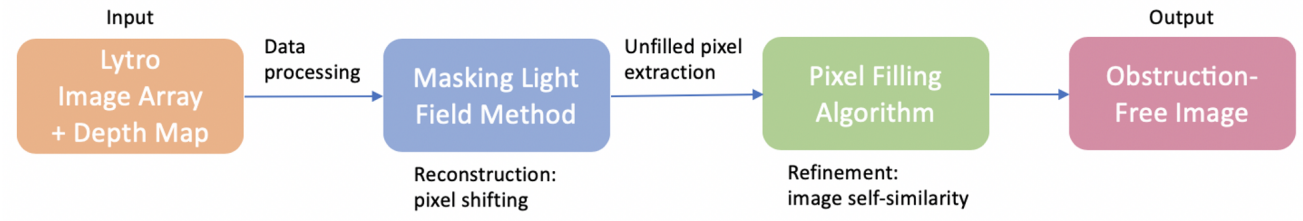

Figure 1: Our obstruction-free image processing pipeline

pixels; 2. region filling and object removal by exemplar-based image inpainting proposed by Antonio Criminisi et al [CPT04] (we call it "region filling algorithm" (RFA) for short in this report).

Firstly, as illustrated in Figure 3, we extract the unfilled pixels left by the MLFM (they would generally be more black than other well-constructed pixels because of MLFM). The main idea is that we set a threshold for the pixel values, and then values below the threshold would be classified as unfilled pixels. We construct a mask to distinguish unfilled pixels from others. Furthermore, consider that if we only use pixel values to distinguish the unfilled pixels, the well-constructed darker pixels in the images would be easily misclassified. So we set a relatively strict threshold and extend those extracted regions by an amount determined by a hyper-parameter "extension". Setting good hyper-parameters would yield better results (refer to the section Results and Analysis). We implemented this pixel extraction part by Matlab.

Figure 3: Illustration of unfilled pixel extraction: (a) Image after MLFM; (b) Extract unfilled pixels by a strict threshold; (c) Extend the regions

Secondly, we used python to implement RFA [CPT04]. The main idea of this algorithm is to mask out the unwanted region, and then fill this region with the most similar patch in term of sum of square difference in the covered area (regions that were not masked out), based on the priority of each pixel at the border of the masked region (see Figure 4). Each pixel at the border of the region would be assigned a priority P, which is a combination of confidence C and the data term D. The confidence term is calculated as the ratio of covered area and total area of the patch centered at the pixel at the border. The patch size is determined by a hyperparameter "window size". The data term is a function of the strength of isophotes hitting the border which can be calculated by the normal vector at the position and the isophote is computed

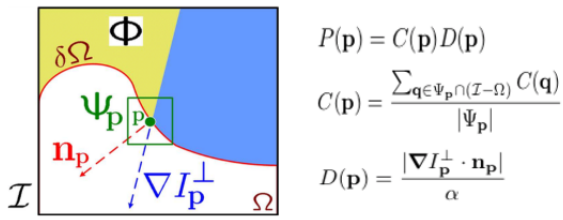

Figure 4: Illustration of the region filling algorithm proposed by Antonio Criminisi et al [CPT04]. The white region Ω (including the red contour) is the masked region, the red contour $\delta\Omega$ is the border of the masked region, the rest of the image covered by the original pixels is denoted with Phi, and the entire image is denoted with I. p is a point at the border of the masked region, Ψ_p denotes the patch centered at p with a region determined by a hyper-parameter window size. n_p is the normal to the border of Ω and $\nabla I_p^\perp$ is the isophote (direction and intensity) at point p.

as the maximum value of the image gradient in the patch. This term would have the advantage of joining disconnected lines. The process would repeat until the whole unwanted region has been covered.

Results and Analysis

In Figure 5, we show three images that we used for testing the pipeline. We can observe that after MLFM, the intersections of the fence are still black because of the lack of information on those regions. But after pixel filling refinement, most of regions are filled with more natural colors and patterns, and the entire image tends to have less unknown blocks. We chose those three sets of images mainly because they have different characteristics. First image has a uniformed color foreground with relatively constant depth from the camera. Second image was taken by ourselves, with various foreground depth and complex shadowing on the fence. Third image contains a continuously changing foreground depth. The qualities of their depth map also vary. Those three sets of images to some extent show that our algorithm is robust to deal with various cases.

Good hyper-parameters can improve the final results. We can see from Figure 5 that there are still some undesirable black pixels in the images. In our pipeline, we mainly have

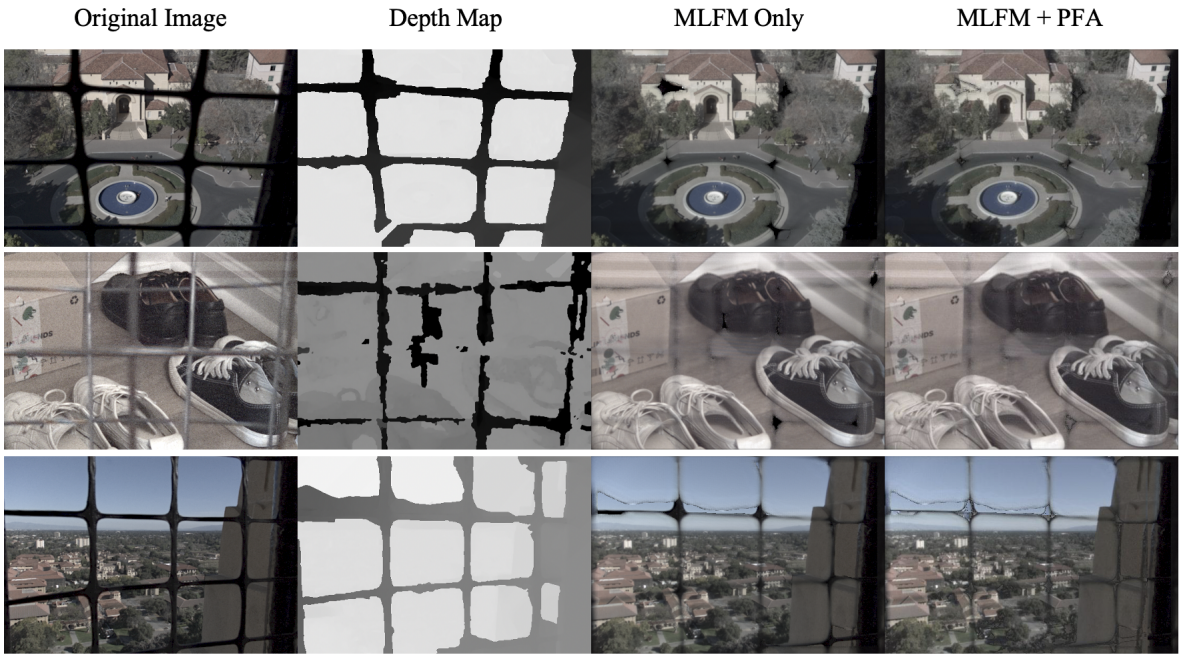

Figure 5: Final Result Comparison

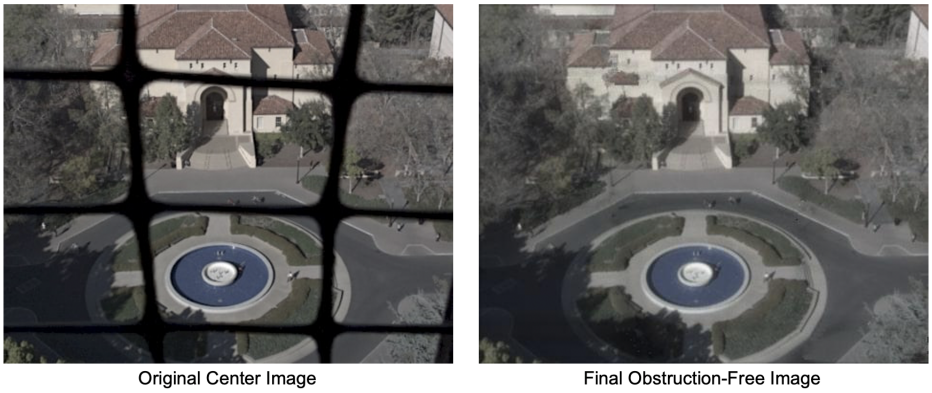

Figure 6: Our final fine-tuned result

Models	MLFM	RFA	Combined Model
Pros	Short runtime Stable and realistic	Only need 1 image + 1 depth info No unfilled pixels	Short runtime Better visual reconstruction
Cons	Complex image pre-processing Visual imperfections (unfilled pixels)	Unstable Long runtime	Still has small artifacts

Table 1: Pro and cons of different models

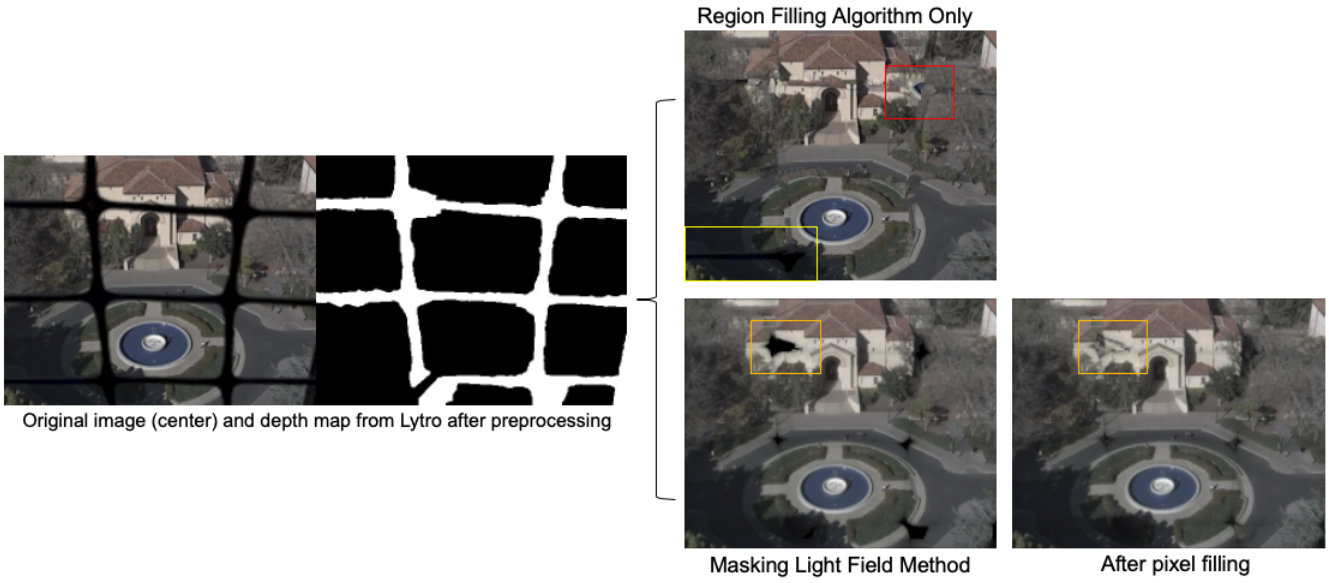

Figure 7: Comparison of different methods

three hyper-parameters: 1. threshold for extracting unfilled pixels, 2. extension for the extracted regions, 3. window size of the patch in RFA. The first two hyper-parameters determined the quality of estimation of unfilled pixels which influences the performance of RFA: if we extract too many pixels, it would cause the RFA to run longer and make it harder to fill the regions (more unknown regions would cause more uncertainty and difficulty of finding reasonable similar regions); if we extract too few pixels, there would be more unfilled black pixels left, resulting in more obvious artifacts. Obviously, the window size influences the comparing and filling part of RFA, which in practice is set to be slightly larger than the largest distinguishable texture element [CPT04]. We did more fine-tuning and got a more desirable obstruction-free image in Figure 6.

Evaluation

We can see from Figure 7 that if we only use the region filling algorithm, the result might contain very unrealistic parts (e.g. part of the center blue circle was shown by the building as indicated in the red box), and the obstruction that was not indicated by the depth map would never be removed (e.g. the bottom left part of the fence stayed the same as indicated in the yellow box). If we only use MLFM, the scene is realistic but there can be large unfilled region as indicated in the orange box. But after filling the lost pixels, the image looks better. Additionally, as can be seen from Table 2, compared with MLFM and RFA, our approach greatly decreases the runtime while improving the overall image quality.

Models	MLFM	RFA	Combined Model
Approx. Runtime	70s	5000s	500s

Table 2: Runtime Comparison

We also list the pros and cons of each method in Table 1. As for the MLFM, it has short runtime with stable and realistic results, but it needs complex image processing and would cause visual imperfection due to the unfilled areas. As for RFA, it only needs an image with corresponding depth map, and it would not have unfilled pixels, but the quality is unstable depending on the image configuration and hyper-parameter choices, and it also has much longer runtime. Our final pipeline combines the advantage of both algorithms, which achieves short runtime and better final image quality, though it can still have small artifacts.

Discussion and limitation

As stated earlier, by using the depth map, our method does not have any constraints on the type of fence, except for a clear threshold that can tell the foreground and background apart. However, we need a relatively precise depth map for

foreground estimation, and this problem can be set as another individual project as well in the future. By using the auto-generated depth map from the light field camera, we inspected some inconsistency between the depth map and the original image, especially for estimating the foreground regions. Some of the foreground estimation was more coarse than we thought and as a result, many useful background information were masked out. To resolve this, we may need to involve machine learning algorithm to further assist with the depth map calculations.

Another limitation is that our current pixel extraction mainly considers separate pixel values, which can cause problem when the image itself contains a large area of darker pixels. It would be a good idea to come up with a more reasonable algorithm based on properties other than each pixel value. Additionally, due to the limitation of our dataset, we did not have many quantitative evaluation metrics.

Future work

In the future, for optimizing the regions that can be well-constructed, we are planning to build non-integer shift methods to reconstruct the background image as precise as possible. We will also continuously refine the pixel filling algorithm, and search for more advanced algorithms to deal with unfilled regions.

Furthermore, if time permitting, we would like to capture image pairs (with and without occlusions) by ourselves to allow for more quantitative evaluations, such as MSE (pixel difference), PSNR (image fidelity), SSIM(structural similarity) and S-CIELAB [ZW+96] (spatial andchromatic encoding of the image by the human eye).

Last, we can investigate more about our algorithm's robustness by running experiments on images with regular or irregular (with some deformation), thin or thick (can be measured by relative coverage) occluders.

Conclusion

As a conclusion, we use pixel location calculation algorithm based on masking light field method to reconstruct the obstruction-free image, and furthermore, we utilize pixel filling algorithm based on the region filling algorithm to fill areas where the real scene information is not well captured by the light field camera or shifting calculation for visual refinement. In general, our approach has certain level of robustness, short runtime, and improves light field obstruction-free image quality.

References

[ZW+96] Xuemei Zhang, Brian A Wandell, et al. "A spatial extension of CIELAB for digital color image reproduction". In: *SID international symposium digest of technical papers*. Vol. 27. Citeseer. 1996, pp. 731–734.

[FS03] Paolo Favaro and Stefano Soatto. "Seeing beyond occlusions (and other marvels of a finite lens aperture)". In: *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* Vol. 2. IEEE. 2003, pp. II–II.

[CPT04] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. "Region filling and object removal by exemplar-based image inpainting". In: *IEEE Transactions on image processing* 13.9 (2004), pp. 1200–1212.

[Hay+06] James Hays et al. "Discovering texture regularity as a higher-order correspondence problem". In: *European Conference on Computer Vision*. Springer. 2006, pp. 522–535.

[Liu+08] Yanxi Liu et al. "Image de-fencing". In: July 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587493.

[Gu+09] Jinwei Gu et al. "Removing image artifacts due to dirty camera lenses and thin occluders". In: *ACM Transactions on Graphics (TOG)* 28.5 (2009), pp. 1–10.

[MLY13] Yadong Mu, Wei Liu, and Shuicheng Yan. "Video de-fencing". In: *IEEE Transactions on Circuits and Systems for Video Technology* 24.7 (2013), pp. 1111–1121.

[McC14] Scott McCloskey. "Masking light fields to remove partial occlusion". In: *2014 22nd International Conference on Pattern Recognition*. IEEE. 2014, pp. 2053–2058.

[Xue+15] Tianfan Xue et al. "A computational approach for obstruction-free photography". In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), pp. 1–11.