Occluded Edge Detection in Light Field Images for Background Removal

Kevin C. Boyle
Department of Electrical Engineering
Stanford University
Stanford, CA
kcb76@stanford.edu

I. Introduction

Standard edge detection or foreground/background separation techniques, such as Otzu's method, require color or intensity differences between the background and regions that need to be separated. For example, green screens are routinely set up as the background in a scene so that there is a clear difference in color between the background and foreground. In a standard 2D image of a scene with a green screen, digital processing can be performed to label every pixel that matches the green screen as part of the background and can be replaced with a corresponding pixel in some other image to change the background of the original image.

Light field imaging aims to capture the complete 4D description of an entire scene, as opposed to the 2D slice of the light field that a standard camera captures. This 4D structure contains depth information about the scene and an important problem in light field research is creating a depth map from the 4D light field. The current state of the art in depth estimation involves sophisticated optimization problems and iterative algorithms. While many of these solutions produce accurate and dense depth estimates then also require significant computational resources [1].

Consider the problem of separating the foreground from the background to imitate the presence of a virtual green screen. A good depth estimation in the scene could provide a metric to separate the foreground and background: simply threshold the depth map for the target green screen location and apply it to the desired viewpoint. This simple technique relies on having an excellent depth map for the whole image, and the corresponding computational cost. Rather than invest all this effort in a good depth map, we can also consider the edges from the pixel intensities in one image and compare these to a more rudimentary depth estimate. This allows foreground/background segmentation across depths without computing a robust depth map.

In this project I implement a simple depth estimation technique using 4D light field data and enhance the foreground/background segmentation performance by incorporating edges found with standard edge detection on the center image of the light field. The end goal is to have a real-time interactable depth segmentation tool for light field images that users can use to perform qualitative virtual green screen techniques. Ease of Use

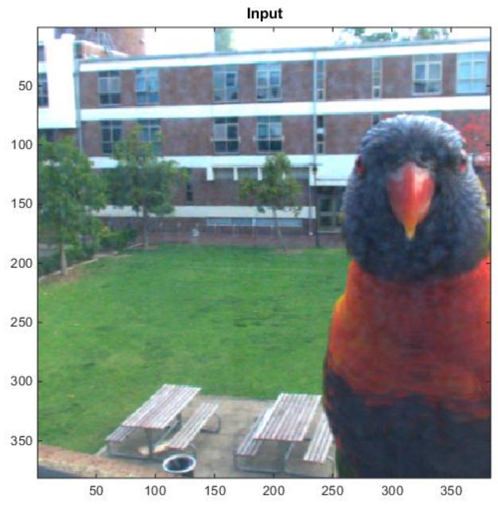


Fig. 1. Central image from a light field of a lorikeet on a window ledge overlooking a distant courtyard.

II. Depth Estimation from Light Fields

The first step in the background removal task is to produce a dense depth estimate from the light field. Throughout this report the working 2D image that will eventually be displayed to the user will be the center image of the light field, such as the one in Fig. 1. That is, for the standard light field 4D representation described in [2], the center view will be from the center of (s,t).

Previous work in the field describes how the orientation of the plane passing through each light field sample is related to the depth of light source at that point [2]. This orientation is related to the 2D gradient in each of (s,u) and (t,v) planes. This thus yields two gradient estimates for each color channel, for a total of six estimates of the depth, each with a corresponding confidence as measured by the magnitude of the gradient. The poor confidence for most points limits the achievable density of a depth map produced in this way. Flat, solid color surfaces with no texture do not have large gradients, while textured edges do. In order to produce a dense depth map, and combine the six estimates across the two planes and three color channels, we can weight the gradients as follows [3].

$$P_z = \frac{\sum \omega_{su} sgn(L_s) L_u + \sum \omega_{tv} sgn(L_t) L_v}{\sum \omega_{su} ||L_s|| + \sum \omega_{tv} ||L_t||}$$

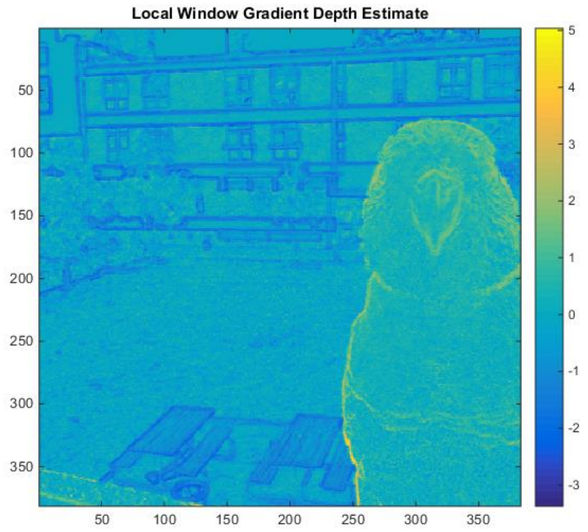


Fig. 2. Depth map estimate of the light field from Fig. 1, where the labelled value represents the slope of the gradient of the plane passing through that light field sample.

Here, Ls, Lu, Lt, and Lv refer to the gradient in the corresponding direction and ω_{su} is a gaussian window. This window incorporates neighboring gradient estimates to produce a dense depth map with reasonable confidence values (note these are captured in the denominators). I use a relatively large 10x10 gaussian window to ensure good density and rely on the detected edges later to recover sharp lines around foreground/background edges. Fig. 2 shows the depth estimate produced for the light field with center image as shown in Fig. 1.

A. *Calibration*

The depth estimate we have constructed so far is not in actual depth units, rather it is the slope of the gradient estimate, which is proportional to the actual depth. For qualitative foreground/background segmentation, we do not strictly need the depth units, and can instead work with the slopes. If actual depth units were required, the light field imager would need to be calibrated, which is discussed in depth elsewhere [3].

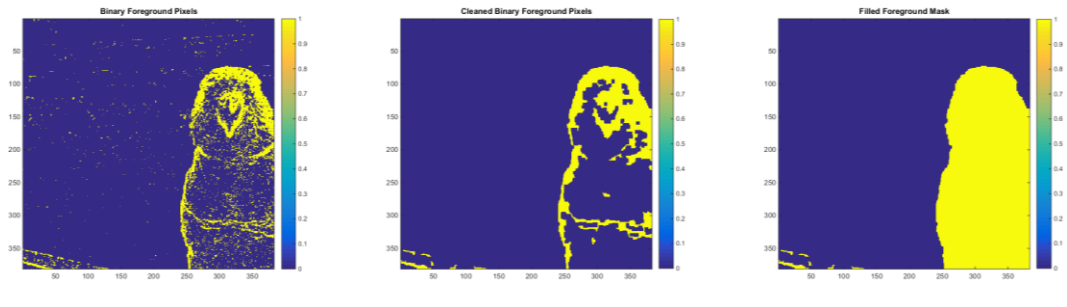
III. Foreground Segmentation

Now that we have a depth map estimate to work with we look to separate the foreground from the background and substitute an image of our own onto the background.

A. *Depth Threshold*

As discussed earlier the simplest way to achieve this is to simply threshold and binarize the depth map and use the result as a mask to clip out the foreground from the center image. However, with the gradient-slope algorithm we used to make the depth estimate we require a small amount of additional processing. It is easily seen in Fig. 2 that the depth map is not uniform or varied enough for thresholding alone to work.

We nevertheless begin by thresholding according to a qualitative user-selected gradient threshold and binarize the pixels in the depth map that remain in front of this threshold as in Fig. 3a. Due to the imperfect depth estimate we are using, there are noisy background depth pixels and the foreground edges are not necessarily fully filled and connected. In the specific sample image we are using here there is also a dirty window pane in the extreme foreground which may be contributing to the noisy estimate, but these erroneous pixels will in general be present regardless. This is easily overcome with small region removal to remove all the small specks below a threshold (about 20 pixels in this case) and a mild (3x3) morphological closing operation to ensure the foreground edges are connected, which results in Fig. 3b. All the bright (yellow) pixels have been identified as foreground pixels, but these are clustered mainly around the edges where there were sufficient gradients in the light field for a high-confidence depth estimate. In order to assign the remaining enclosed regions, we region label and look at each individually. For each dark region we take the mean of all the depth estimates for the corresponding pixels and generate one average depth estimate for the whole region. We then compare this to the selected threshold to determine if



(a) (b) (c)

Fig. 3. Steps in foreground segmentation after the thresholding operation: a) binarized pixels from the depth map above the threshold, b) small regions removed and closing operation applied, c) region filling on areas identified as foreground.

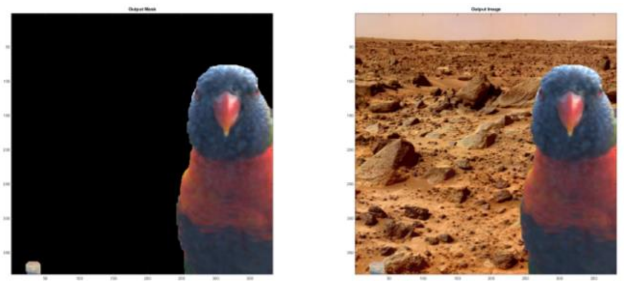


Fig. 4. Foreground mask applied to the center image from Fig. 1 and a new image superimposed onto the background.

the whole region is in the foreground or background and assign the pixels accordingly. Specifically, we compare the average depth estimate to 20% of the user-selected threshold, to account for the lower gradients in regions without edges. This finally leaves us with the binary mask in Fig. 3c that shows the pixel locations of the foreground with respect to the center image.

The final step is to clip out the foreground pixels of the center image and replace the background with an image of our choosing. For more general performance across many images an additional set of opening and eroding morphological operations can be applied. In Fig. 4 we have done just that to successfully put the first lorikeet on Mars. The whole thresholding operation is carried out in real-time after the depth map is formed (this only takes a few seconds up front) and is controlled with an interactable GUI that sets the depth threshold used at the beginning of the process.

This basic thresholding algorithm works well in many cases, especially those with a large, distinct foreground area in front of a distance background. In these cases the foreground pixels can be aggressively thresholded, thus removing the background entirely, and still leave enough information to close up the edges in the foreground mask. However, this method is not sufficient for cases such as that in Fig. 5 which has a much flatter distribution of depths and many regions of low-confidence estimates. Here, to select the whole window ledge and both birds we must lower the threshold so far that the background scene bleeds in. The next section improves upon the simple thresholding to correct this.

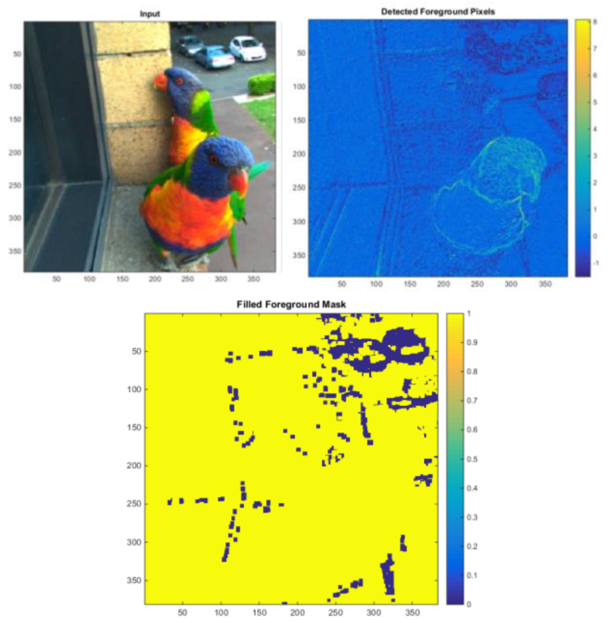


Fig. 5. (top row) center image of light field and the gradient-slope depth estimate, (bottom row) failed foreground map produced by the threshold only algorithm to segment the birds and windowsill from the background below.

B. *Occluded Edge Detection*

To supplement the depth map estimate we look to the edges in the center image that we are ultimately applying the segmentation to. The edges from pixel intensities in this center image (consider Fig. 5) should ultimately correspond to either unoccluded edges, such as the coloring pattern on the lorikeet or the texture on the brick wall, or to occluded edges, such as the boundary between the bird and the distant driveway. Occluded edges that exist above the desired threshold are exactly where we want to segment the image, so we turn now to incorporating these edges into our algorithm.

We begin again with the depth map estimate just as before, which can be seen in Fig. 5, and threshold it according to a user-selectable value. Then, we apply canny edge detection to the grayscale center image, resulting in Fig. 6a. We are only interested in the edges that can be considered to be in the foreground so we remove any edge pixel that is not at least 20%

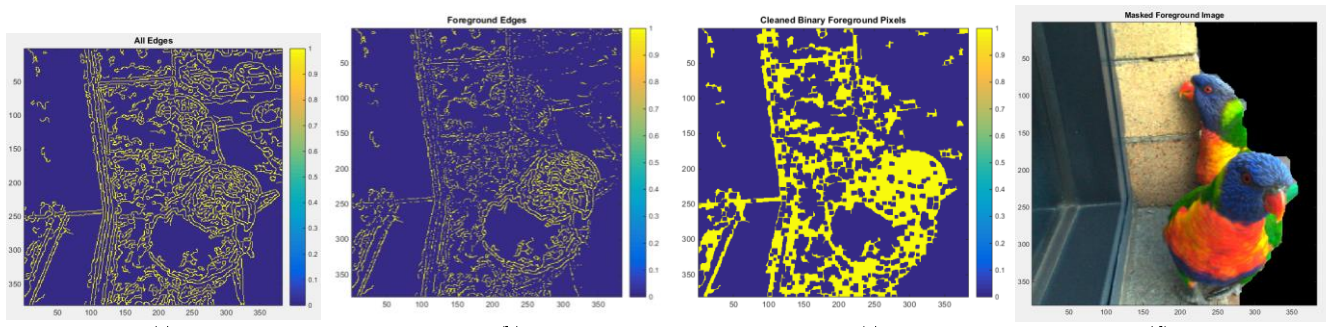
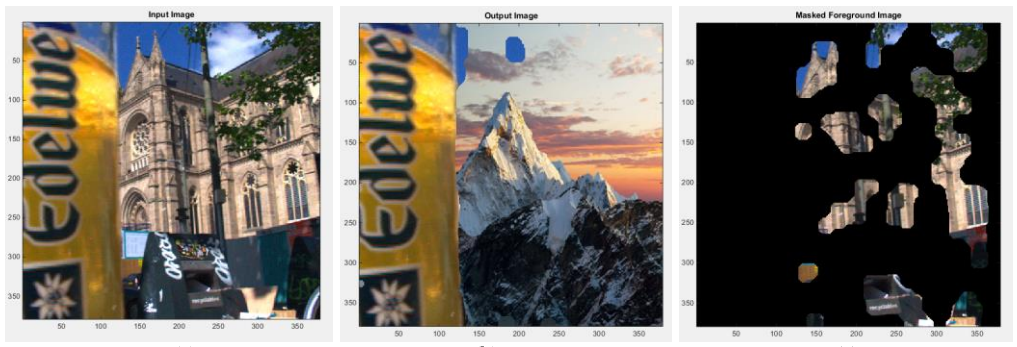


Fig. 6. Intermediate steps in the edge detection algorithm, (a) canny edge detection on center image, (b) foreground edges that agree with the depth map, (c) cleaned foreground edges to be filled to create mask, (d) masked foreground image segmenting the windowsill from the ground below.a



(a) (b) (c)

Fig. 7. Example of threshold-only versus edge detection performance in a specific case, (a) original input image, (b) threshold-only approach with a background applied, (c) failed edge detection foreground mask created by low confidence in the foreground region and small edge-detected regions in the background that are sensitive to the depth estimate noise.

greater than the user selected threshold, again the 20% is to account for edges that cross low-confidence depth estimate regions without throwing them out too quickly. This gives us foreground edges as in Fig. 6b, which we treat the same way we did the detected foreground pixels from the previous section in Fig. 3a. We continue on from here exactly as before, applying region filling and morphological processing to create finished edges as in Fig. 6c. This mask is then region labelled and compared to the depth estimate to fill in the foreground regions and can produce a final image as in Fig. 6d. This result is clearly much improved over the simple thresholding from Fig. 5 and generally works well in cases where the thresholding-only method fails.

There are cases where the thresholding-only method is superior, especially when the depth distribution is very binary and there are lots of edges in the background. In these cases the thresholding works reliably with the large difference in foreground and background depth estimates, but the edge detection is bound to find some small regions in the background that appear to be foreground. The algorithm here, that looks at the mean of the depth estimate in enclosed regions after thresholding, weights these small noisy regions as more likely to be foreground than larger regions and consequently fails. Future improvements on the region-by-region foreground probability estimates could alleviate this problem. Fig. 7 shows an example where the edge detection method fails compared to the thresholding-only method.

IV. Conclusions

The two depth segmentation algorithms discussed here both run in real-time and allow the user to interactively segment the image into background and foreground regions and substitute in a new background image. These two methods work well in complimentary cases and future work to improve the edge detection method or fuse the two results of the two methods could see substantial improvements. State of the art work on depth estimation from light fields can generate significant improvement in performance as well, but at the cost of greatly increased computation time for the initial depth estimate. Some depth estimation methods already include edge detection in the center image [4].

Acknowledgment

This project made use of the Matlab Light Field Toolbox v0.4 by Donald Dansereau. The toolbox has a number of useful functions for loading and processing light field data from commercial Lytro cameras and other datasets. The toolbox also contains some sample light field data, such as the lorikeet photos used in this report.

Additional light field datasets were taken from the Lytro First Generation Dataset available online at https://www.irisa.fr/temics/demos/lightField/index.html [5].

References

[1] H.-G. Jeon, et al, “Accurate depth map estimation from a lenslet light field camera,” *IEEE Int. Conf. Computer Vision Pattern Recognition*, 2015.

[2] D. Dansereau, and L. Bruton, “Gradient-based depth estimation from 4D light fields,” in *Proc. 2004 Int. Symp. Circuits Systems*, 2004.

[3] D. Dansereau. “Plenoptic signal processing for robust vision in field robotics,” Ph.D. dissertation, Sch. Aerospace, Mech., Mechatronic Eng., Univ. Sydney, Sydney, Australia, 2014.

[4] T.-C. Wang, et al, “Depth estimation with occlusion modeling using light-field cameras,” *IEEE Trans. Pattern Analysis Machine Intelligence, 2016.*

[5] A. Mousnier, E. Vural and C. Guillemot, "Partial Light Field Tomographic Reconstruction From a Fixed-Camera Focus Stack", *IEEE Trans. on Image Processing*, submitted, 2015..