Depth from Defocus for Misaligned Mobile Photos Subject to Parallax Effects

Robert Mahieu
Department of Electrical Engineering
Stanford University
rmahieu@stanford.edu

***Abstract*—This paper describes and implements an attempt to solve the problem of determining depth from defocus when given a focal stack (set of photos taken at different focuses) created from photos captured with a hand-held camera with unknown calibration. Although the photos may initially suffer from significant parallax due to camera/hand movement during capture, the image alignment steps explained in Section III are shown to be capable of compensating for such issues. Relative depth results are computed quickly, though suffer from some shortcomings around depth edges as discussed in Section VII. The final computed depth map is also shown to be sufficient for rendering visually pleasing synthetically refocused photos of the scene.**

I. Introduction

Obtaining the depth map of a scene has many benefits in photography, such as providing the ability to refocus a photo after-the-fact or, most notably, enabling the camera to auto-focus by setting its focal plane to the depth of a specific object. However, with photographs taken only from a single viewpoint, and without using any specialized depth sensors, the problem of determining scene depth becomes difficult.

When capturing a photograph of a scene, how focused an object becomes in the resulting photo is a direct result of its distance from the focal plane of the camera. The farther the object is from this focal depth, the more blurring that occurs on the camera sensor. Consider, then, a case in which we take a photo with the focal plane set at the depth of the closest object in the scene. If we possessed a copy of this photograph with every object in focus, it might then be possible to estimate the size of the blur kernel (point spread function) that produced the actual blurry photograph, thus enabling us to determine a measure of relative depth of each object in the scene.

The notion described above is the main idea behind depth-from-defocus (DfD) techniques, which attempt to reconstruct a depth map of the scene using a set of photographs, referred to as the *focal stack*, which are each taken of the same scene with the camera set to different focal depths. Throughout this paper we assume that the first image in the stack is captured with the focal depth set to the depth of the closest object in the scene and the last captured with the focal depth set to the depth of the furthest object.

Utilizing such a DfD technique on hand-held mobile cameras, however, becomes challenging due to the inevitable motion of the camera that occurs between the capture of each photo in the focal stack from involuntary hand movement. In this scenario, the photos will likely be subject to significant parallax effects, causing some objects that are behind others to be captured in one image but not show up at all in others. This then leads to error within depth computation as corresponding pixels in each image in the focal stack do not represent the same object. Furthermore, we assume that calibration of the hand-held camera is unknown, so we cannot leverage aperture, focal length, and focal depth values in our solution.

While realignment through affine image registration may work well enough for sufficiently small motion, the effects of parallax caused by large motion are too substantial to correct for using only this method. Instead, this paper describes a more robust technique that uses a combination of affine transformation and optical flow fields to register images in the focal stack before estimating depth.

The complete proposed system takes as input a focal stack generated from a hand-held camera. The first step is to stabilize the photos relative to the first image in the stack by correcting for magnification differences and parallax between all the photos. Using this aligned focal stack, then compute an all-in-focus photo by formulating this task as a Markov Random Field (MRF) optimization problem. Finally, using the all-in-focus image, determine the size of the blur kernel at each pixel that produces the most similar results to the images in the focal stack. Using the results for the photos in the stack with the

closest and furthest focal planes, we can then use the kernel size per pixel as a measure of relative depth for the objects in the scene.

II. Related Work

Most work into depth-from-defocus techniques assumes camera calibration parameters are known, which provides a simplification from the problem investigated within this project. However, some work has been done under these constraints. Ens and Lawrence [1] proposed an iterative matrix based method, using only two images of the same scene taken at different focal depths, to solve for the defocus operator. They used a precomputed table of one of the photos filtered by different defocus operators to select the operator for each pixel's local neighborhood which minimized difference with the other photo. This showed promising results and essentially describes the technique utilized in the system used here in this paper for determining relative depth.

A recent extension of this work done by Mannan and Langer [2] formulates the problem in 2D and in the filter space with explicit non-negativity and unity constraints. Their optimization problem is solved with Quadratic Programming. This approach makes no assumptions about the type of point spread function present, though it requires use of special calibration and test images and cannot use photos of a regular scene.

Furthermore, Suwajanakorn et al. [3] recently proposed a pipeline for determining, from a focal stack produced with a hand-held camera, not only depth from focus but also aperture size, focal length, and focal depths (up to an affine ambiguity) for each photo by formulating an optimization problem that jointly solves for all parameters. This work describes the focal stack stabilization as well as the all-in-focus image generation techniques utilized in the system in this paper.

III. Image Alignment

To compensate for camera movement during capture of the focal stack, realignment of the photos must be done as a first step. Magnification changes are first reconciled by matching all images in the focal stack to the first image with affine registration, which solves for an affine transformation that best matches the images. This project uses the Lucas-Kanade inverse compositional iterative registration algorithm from the Image Alignment Toolbox [4] to determine each individual transformation between consecutive photos, then multiplies these transformations appropriately to get a transformation for each photo that matches it to the first photo. For a set of images of size 363x653 pixels, this step took about 1.5sec per image. We set the algorithm to use 50 iterations at only a single level.

To then handle the parallax discrepancy between photos, we solve for a dense correction field using optical flow. Let I_i denote photo i in the focal stack, and let $F_i^j : \mathbb{R}^2 \to \mathbb{R}^2$ denote the optical flow field matching I_i to I_j. To complete this matching step, first the 2D optical flow field between each consecutive image in the now affine-aligned focal stack, $F_2^1, F_3^2, F_4^3, ..., F_N^{N-1}$, is determined. We compute optical flow fields using the MATLAB implementation by Ce Liu [5] which was able to produce very good results at around 4sec per image.

To then get the flow field for each image that aligns them to I_1, we concatenate the flow fields utilizing the following warping function, which is formulated as a way to align an image I according to the flow field F:

$$W_F(I(u,v)) = I(u + F(u,v)_x, v + F(u,v)_y)$$

Where F_x and F_y are the x- and y-components of the flow field respectively. To concatenate flow fields and compute each alignment field relative to I_1, we recursively perform warping of each flow field according to the formula $F_i^1 = F_i^{i-1} \circ F_{i-1}^1$ using the $\circ$ operator defined as $S = F \circ F'$ where:

$$S_x = F'_x + W'_F(F_x)$$
$$S_y = F'_y + W'_F(F_y)$$

Since we already have F_2^1 at the start of this step, we can then compute F_3^1, then F_4^1, and so on according the the formula above. Once we have these concatenated flow fields, we can then perform alignment for each image I_i by computing $\hat{I}_i = W_{F_i^1}(I_i)$. To fill holes in the resulting warped image, our implementation uses a simple linear interpolation, though to get better results we strongly recommend using an inpainting method.

IV. All-in-Focus Image

Using this aligned focal stack, with each image now denoted by $\hat{I}_i$, we can now compute a photo of the scene where every object is in focus. This all-in-focus image is generated by using for each pixel, the image in the focal stack in which that pixel is "sharpest". This then requires us to create a sharpness metric which we define as the weighted sum of the image gradient magnitude, expressed as $\exp|\nabla I(u,v)|$, in a Gaussian neighborhood with isotropic variance σ^2 around our target pixel (u,v). We were able to achieve good results using the Sobel operator for the image gradient computation as well as a standard deviation of 3 pixels for the Gaussian neighborhood.

To actually choose, for each pixel in our all-in-focus image, the index of the best image in the focal stack, we represent the problem as a Markov Random Field (MRF) subject to the following energy function:

$$E(x) = \sum_{i \in \mathcal{V}} D(x_i) + \lambda \sum_{(i,j) \in \mathcal{E}} V(x_i, x_j) \qquad (1)$$

Where the variable x represents a set of assigned indices for all pixels. We represent pixels in the image as the graph nodes, $i \in \mathcal{V}$, and use the edges of the graph, $\mathcal{E}$, to represent a 4-connected relationship between neighboring pixels. We define the unary potential (data cost), $D(x_i)$, of a pixel i being assigned an index x_i as the sharpness metric value computed according to the description above.

We also define the interaction potential (smoothness cost), $V(x_i, x_j)$, of pixels i and j being assigned indices x_i and x_j respectively as the absolute difference between the index values: $|x_i - x_j|$. This term is also regularized by the parameter λ which controls the importance of obtaining smooth results. One of the primary benefits of formulating this problem as an MRF is the ability for us to impose this smoothness constraint on the result. As we have it defined, this smoothness constraint functions as a linear total variation prior. Figure 1 illustrates this effect.

The optimization problem of minimizing the energy function in Equation 1 is solved in our implementation using the α-expansion graph cuts algorithm implemented in MATLAB by Veksler and Delong [6][7][8]. For a focal stack of 13 363x653 images, the all-in-focus image was generated in about 16sec.

V. Depth Estimation

Now that we have our all-in-focus image, which we will denote as $\hat{I}_{AF}$, we can now proceed to actually determine a measure of depth for the objects in our scene. We do this for a given image in the focal stack, $\hat{I}_i$, by comparing a local neighborhood around each pixel to the corresponding region in several copies of the all-in-focus image convolved with blur kernels of various sizes and then selecting the kernel that produced the image that matches best.

The first part of this step is to create the uniformly blurry images from $\hat{I}_{AF}$. We assume a disk point spread function and therefore convolve $\hat{I}_{AF}$ with several disk-shaped blur kernels of increasing radius r. In our implementation we used radii from 0.25px to 6.50px in increments of 0.25. This then generates a set of images which we can refer to as the *blur stack* and denote each image as B_r where r signifies the radius of the blur kernel that produced it. Note that we also have $B_0 = \hat{I}_{AF}$ as part of the stack. Additionally, we grayscale the all-in-focus image prior to these computations and thus produce single-channel blurred images.

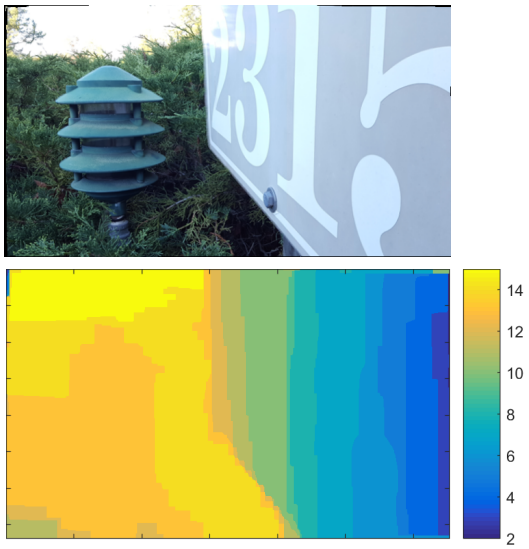


Fig. 1: Example of smooth results possible using this method for all-in-focus image creation. *Top)* All-in-focus image. *Bottom)* Index map generated from MRF optimization.

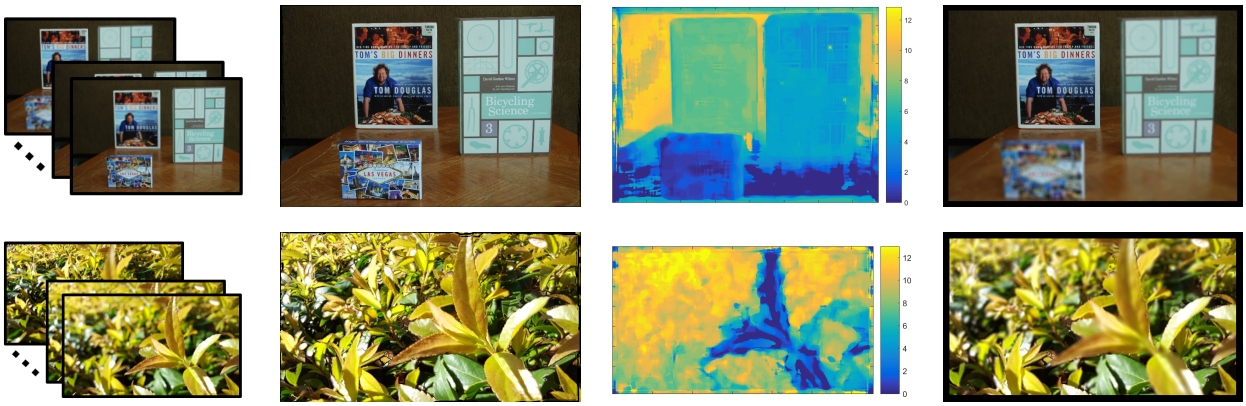
Next we attempt to find the closest match among the blur stack images for each pixel in some reference image I, which we assume has been grayscaled. To do this, we compute a difference map, $D_r : \mathbb{R}^2 \to \mathbb{R}$, for each kernel radius r by taking the sum of absolute difference between the reference image I and the blurry image B_r over a Gaussian neighborhood around each target pixel. This is reflected in the following equation:

$$D_r(u, v) = \sum_{(u', v')} w(u' - u, v' - v) |I(u', v') - B_r(u', v')|$$

Where (u, v) represent the coordinates of the target pixel and (u', v') refers to all pixel coordinates in the neighborhood. $w(u, v)$ refers to a Gaussian weighting function centered at (0,0). In our implementation we obtained good results using an isotropic standard deviation of $\sigma = 11$px for the Gaussian, though this parameter is somewhat image-dependent and should be tuned for images of different sizes and contents.

To then choose which kernel size to select for each pixel, we compute what we refer to as a *radius map*. This is done by determining the kernel radius that had produced the best-matching image in the blur stack:

$$R(u, v) = \delta \cdot \operatorname*{argmin}_r D_r(u, v) \qquad (2)$$



(a) Focal Stack (b) All-in-Focus (c) Complete Radius Map (d) Refocus

Fig. 2: Examples of results generated from hand-held camera photos captured during large motion. Top focal stack images taken from [3]. (a) shows the input to the system, (b) shows the all-in-focus image generated from the input, (c) shows the relative depth map computed, and (d) gives one example of possible refocusing that can be achieved using the computed depth.

Note the presence of scale factor δ which works to undo the magnification correction we originally applied in the affine alignment step. This is important because without this scaling, our resulting radius value would not properly represent the actual object depth at that pixel. To compute δ our implementation simply uses the inverse of the average of the s_x and s_y scale terms within the affine transformation matrix T we applied to the reference image during alignment:

$$\delta = \left(\frac{T_{11} + T_{22}}{2} \right)^{-1}$$

Because the $\hat{I}_1$ photo should have its focal depth set to the depth of the closest object in the scene, the radius of the blur we detect at each pixel in the image will directly correlate to the depth of the object at that location. Similarly, the $\hat{I}_N$ photo should have its focal depth set to the farthest object in the scene, so the blur radius will negatively correlate to scene depth. Therefore, if we perform this radius-map computation step using both $\hat{I}_1$ and $\hat{I}_N$ as reference images (creating maps R_1 and R_N respectively), we can finally construct a complete radius map representing the relative depth of the scene by combining the two results according to:

$$R_{complete} = R_1 + (\max R_N - R_N)$$

This can now function as a depth map for the scene.

VI. Refocusing

To demonstrate the use of this result, we can produce synthetically refocused photos of the scene. This can be done for some specified relative depth d_{rel} (typically between 0 and 13 in our case since our max test radius was 6.5px) by convolving the all-in-focus image $\hat{I}_{AF}$ with a spatially varying disk blur kernel of radius:

$$r(u, v) = |R_{complete}(u, v) - d_{rel}|$$

This is a very time consuming process—we found that each refocused image took around 41sec to generate—but does produce some interesting results.

VII. Results & Discussion

Some selected results are illustrated above in Figure 2. We find that the algorithm generally produces strong results, and seems to perform better when there are large areas in the scene that lie at the same depth. This is likely a result of the size of the Gaussian neighborhood we utilize during difference map computation. Even so, in more complex scenes such as the leaves photoset shown on the bottom of Figure 2, where the photographs were taken with not only hand movement but also with the leaves themselves shaking in the wind, we are still able to produce visually pleasing results during synthetic refocusing (Figure 2d).

Our implementation was done fully in MATLAB, and processed a focal stack of 13 368x653 pixel photos in about 2.5min plus around 41sec per refocused image rendered. This is a significant decrease from the time taken by [3], which took over 9min to complete similar steps, but at a noticeable loss in quality.

One of the major shortcomings of our implementation comes from the fact that during optical flow warping, holes in the resulting image are filled by simple linear interpolation between pixels. This certainly contributes

to the speed of our pipeline, but we suggest that future implementations use some form of inpainting instead. The interpolation artifacts are very noticeable when large parallax is present and cause problems at the depth boundaries. Examples of this issue are shown below in Figure 3.

Fig. 3: Examples of artifacts produced from linear interpolation during optical flow warping. The above images were generated after image alignment.

VIII. Conclusion

The results from this paper show that even with significant hand movement during photo capture on an uncalibrated hand-held camera, it is possible to reconstruct a sufficient measure of depth within the scene. This has potential for use in many applications such as after-the-fact image refocusing, scene reconstruction, or scene understanding. The fact that this technique is possible using photos from mobile devices is a step towards making such technology available to the average consumer.

References

[1] Ens, John, and Peter Lawrence. "An investigation of methods for determining depth from focus." *IEEE Transactions on pattern analysis and machine intelligence* 15.2 (1993): 97-108.

[2] Mannan, Fahim, and Michael S. Langer. "Blur calibration for depth from defocus." *Computer and Robot Vision (CRV), 2016 13th Conference on.* IEEE, 2016.

[3] Suwajanakorn, Supasorn, Carlos Hernandez, and Steven M. Seitz. "Depth from focus with your mobile phone." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

[4] Evangelidis, G. "IAT: A Matlab toolbox for image alignment." (2013).

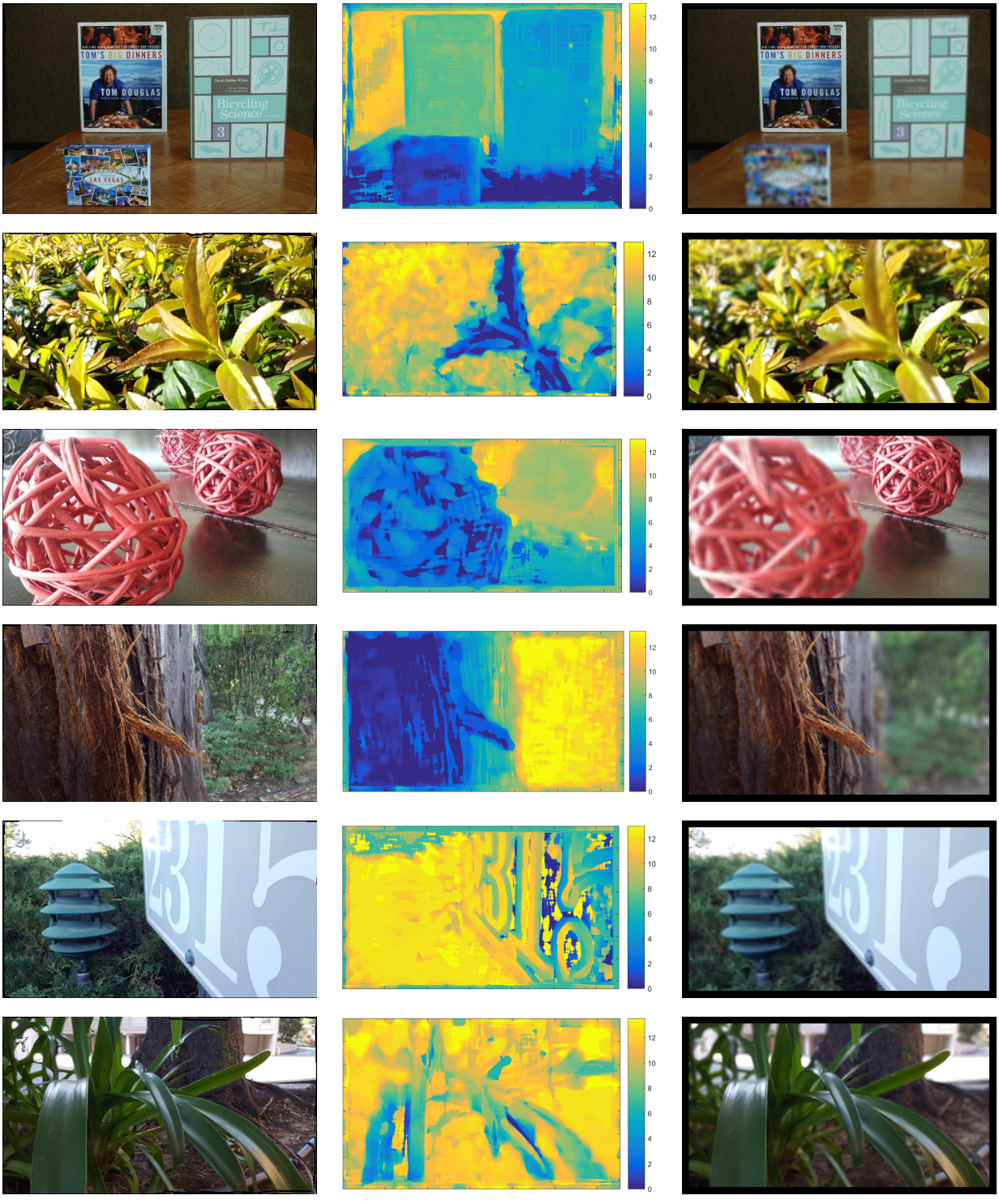
[5] Liu, Ce. *Beyond pixels: exploring new representations and applications for motion analysis.* Diss. Massachusetts Institute of Technology, 2009.

[6] Boykov, Yuri, Olga Veksler, and Ramin Zabih. "Fast approximate energy minimization via graph cuts." *IEEE Transactions on pattern analysis and machine intelligence* 23.11 (2001): 1222-1239.

[7] Kolmogorov, Vladimir, and Ramin Zabin. "What energy functions can be minimized via graph cuts?." *IEEE transactions on pattern analysis and machine intelligence* 26.2 (2004): 147-159.

[8] Boykov, Yuri, and Vladimir Kolmogorov. "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision." *IEEE transactions on pattern analysis and machine intelligence* 26.9 (2004): 1124-1137.

IX. Appendix: Extended Results



(a) All-in-focus (b) Relative depth (c) Refocus example

TABLE I: Experimental results. (a) shows all-in-focus images, (b) shows estimated relative depth maps, and (c) shows refocusing examples.