Extracting the Depth and All-In-Focus Image from a Focal Stack

Shane Barratt Benjamin Hannel
Stanford University
{sbarratt,bhannel}@stanford.edu

Abstract

Modern cameras have a number of imaging parameters (e.g., aperture, focal length, and focus distance) that can be changed quickly between successive image captures. Depending on the specific hardware configuration of the camera, different imaging parameters result in different images, with the primary difference being which depths are in focus. In this project we consider focal stacks, which are a number of images taken of the same scene with different imaging parameters. Given a focal stack, we consider the problem of simultaneously extracting the all-in-focus image, i.e., the image where every depth is in focus, and the depth of each pixel in the image. We frame this problem as an inverse problem, with a particular image formation model, and propose a number of solution algorithms, including gradient descent, alternating minimization, and ADMM. We apply our gradient descent method to simulated images as well as images taken with a Samsung Galaxy S8 Active Android smartphone, and find the performance satisfactory.

1. Introduction

The advent of computer-controlled cameras (e.g., DLSRs, smartphones) make it easy to automatically capture a collection of images with different imaging parameters (*e.g.*, aperture, focal length, and focus distance) in rapid succession. A collection of images captured with different imaging parameters, which we refer to as a *focal stack*, contain much more information about a scene than a single image. For example, a focal stack can be used to approximately extract the depth of each pixel in a scene, since whether or not a pixel is in focus depends on the distance from the camera to the point in the scene. This is impossible to do using just a single image without additional contextual information. In addition, it is possible to approximately compute the all-in-focus image, *i.e.*, the image where every pixel is in focus.

In this project we pose the problem of jointly inferring the depth map and all-in-focus image as an inverse problem, assuming each image is formed as a (spatially-dependent) convolution of the all-in-focus image with a disc-shaped PSF whose diameter depends on the depth of the pixel it is centered on, the aperture diameter, the focal length, and the focus distance (all of which we know except the depth). To render the problem more tractable, we discretize the depth into a finite number of distances, and consider as our optimization variable a probability distribution over these depths for each pixel as well as the all-in-focus image. (We can compute the continuous depth as the expected value of this distribution, and this has the added benefit of an uncertainty measure on the depth.) As a result, our image formation model is biaffine in the all-in-focus image and probability distribution over depths.

The resulting inverse problem has three terms in the objective: a reconstruction loss between the output of the image formation model and the captured images, regularization on the all-in-focus image, and regularization on the depth probability distribution. We focus on the case where the reconstruction loss function and both regularization functions are convex, meaning the inverse problem is a biconvex optimization problem, composed of a convex objective function and biaffine constraints. We propose several methods to approximately solve this problem, gradient descent, alternating minimization, and ADMM. Alternating minimization and ADMM can naturally deal with non-smooth terms in the objective function. All of these methods have the added benefit of warm starting, which make hyper-parameters sweeps and successive application of the method to similar focal stacks fast.

We implement the aforementioned algorithms in PyTorch, a library for computation on multi-dimensional arrays, which allows the algorithm to seamlessly scale to one or more CPUs or GPUs. We represent the image formation model as an abstract (bi)linear operator. With a particular re-ordering of the steps in the image formation model, we can convert it from a number of spatially-dependent convolutions, followed by a weighted average to a number of spatially-independent convolutions, followed by a sum over a particular dimension. This re-ordering is much more efficient on homogeneous computation architectures such as GPUs. We describe this process in more detail in the sequel.

We evaluate our method on both simulated data and images captured using a Samsung Galaxy S8 Active Android smartphone. We find that the method is able to reliably deduce the depth and all-in-focus image, and also runs very quickly, with the image formation model taking roughly 6 milliseconds for a focal stack composed of ten 256 x 256 images, 5 depth discretizations, and a maximum filter size of 11 x 11. In order to perform the Android experiment, we had to perform extensive system identification on the smartphone's imaging system, which we also describe in detail.

Summary. The remainder of the paper proceeds as follows. In §2, we summarize related work. In §3, we describe our image formation model, our inverse problem formulation, and our proposed solution method. In §4, we present our experiments on simulated and Android images. In §6, we conclude the paper with a discussion of our results and possible directions for future research.

2. Related work

Autofocus. The idea of extracting depth and an all-in-focus image from focal stacks is by no means new, and arguably began due to an interest in autofocus, *i.e.*, finding the best focus distance to bring a particular part of an image in focus. To the best knowledge of the authors, the first mention of autofocus in academic literature was due to Horn in 1968 [8], and since then, it has received a lot of attention (see [22] for a survey), and has become a necessary fixture in all DLSRs [17, 11] and smartphones [26, 2].

Sharpness-based approaches. A common approach to the problem of reconstructing depth from a focal stack is to compute the sharpness of each region in the image, find which image in the focal stack that region is the sharpest, and letting the depth of that region be the focus distance of that image in the focal stack. In 1976, Jarvis proposed a variety of sharpness measures, including entropy, variance, and total variation [12, 13]. Follow-up work utilized the fact that there is a smooth gradient of focus as a function of depth, and took into account multiple images in computing the depth [19, 6, 23]. We note that similar techniques have also been used to recover shapes [16].

Inverse-based approaches. Ens & Lawrence were among the first to pose depth from defocus as an inverse filtering problem, and added regularization [4]. Chaudhuri & Rajagopalan wrote a book on depth from defocus using inverse-based approaches [3]. More recently, researchers have focused on depth from defocus in particular on mobile phones [24, 25]; here the images need to be aligned and the underlying parameters of the camera are not known exactly, since there are many phone models. Both of them apply generic nonlinear optimization routines, which are computationally intensive, requiring at least thirty minutes to process a single focal stack. (This precludes their use on actual mobile phones.) We also note that it is still unclear what the best image formation model is to use for the problem of depth from defocus [15].

Depth from other sources. Researchers have proposed extracting depth from other sources. For example, they have proposed to extract depth from a single image, which is sometimes called monocular depth estimation [27]. Modern approaches to this problem include random forests [20] and neural networks [14, 5]. Researches have also proposed approaches for extracting depth from stereo imagery, *i.e.*, one or more cameras taking images of the same scene from different positions or angles [21].

3. Method

3.1. Image formation model

We consider a scene, represented by an all-in-focus image $I \in \mathbf{R}_+^{H \times W}$, from the perspective of a camera. (We work with grayscale images, though the methods described can be applied to color images with minor modifications.) Each pixel in the image is assumed to have a depth, represented by a depth map $D \in \mathbf{R}_+^{H \times W}$. We assume that N images are taken with different imaging parameters, resulting in images $I_1, \ldots, I_N$.

Image formation model. Our image formation model for the focal stack $\hat{I}_i \in \mathbf{R}^{H \times W}$, $i = 1, \ldots, N$, has the form

$$(\hat{I}_i)_{xy} = \sum_{u=1}^{H} \sum_{v=1}^{W} I_{uv} \mathbb{1} \left[\|(u,v) - (x,y)\|_2 \leq C_i(D_{xy}) \right],$$

where $\mathbb{1}(x)$ equals 1 if x is true and 0 otherwise, and C_i denotes the radius of the circle of confusion (in pixels) for a depth d in the ith image of the focal stack, which has, *e.g.*, the form

$$C_i(d) = \frac{1}{2s} \frac{f_i}{S_i - f_i} \frac{f_i}{N_i} \frac{|d - S_i|}{d}, \quad i = 1, \ldots, N,$$

where s is the size of a pixel (in meters), and for the ith image in the focal stack, f_i is the focal length (in meters), S_i is the focus distance (in meters), and N_i is the f-number (unitless).

Depth discretization. In its current form, the image formation model is not a continuous function of D, which we will require in the sequel. To remedy this, we discretize

the depth into K depths $d_1, \ldots, d_K$, and consider a (pixel-dependent) probability distribution over these depths, denoted

$$P \in \Delta = \{P \in \mathbf{R}_+^{H \times W \times K} \mid P\mathbf{1} = \mathbf{1}\mathbf{1}^T\}.$$

We hope that the expected value of this distribution, Pd, is approximately equal to the true depth. In addition, discretization has the added benefit of giving us other statistics on the depth; for example, the variance of the depth at each pixel is

$$\mathbf{var}(P)_{xy} = \sum_{i=1}^{K} P_{xyi}(d_i - (Pd)_{xy})^2.$$

Simplified image formation model. With the depth discretization, the image formation model takes the form

$$(\hat{I}_i)_{xy} = \sum_{j=1}^{K} \sum_{u,v} P_{uvj} I_{uv} \mathbb{1}\left[\|(u,v) - (x,y)\|_2 \leq C_i(d_j)\right].$$

Fast image formation. To compute $\hat{I}_i$, we first compute K depth probability-weighted images

$$P_{xyj} I_{xy}, \quad j = 1, \ldots, K,$$

then convolve each of these images with the (pixel-independent) disc-shaped PSF with radius $C_i(d_j)$, and then sum each of these images. We can implement this in PyTorch using just multiplication, addition, some reshape operations, and 2-D convolution, `torch.conv2d`.

3.2. Inverse problem

Denote the reconstruction loss by

$$L(I, P) = \sum_{i=1}^{N} \ell(\hat{I}, I_i),$$

where $\ell : \mathbf{R}^{H \times W} \times \mathbf{R}^{H \times W} \to \mathbf{R}$ is some (differentiable) loss function. The inverse problem that we seek to solve has the form

$$\text{minimize} \quad L(I, P) + r_I(A_I I) + r_P(A_P P), \qquad (1)$$

with variables I and P, where A_I and A_P are (abstract) linear operators, r_I is the (convex) all-in-focus regularization function, and r_P is the (convex) depth regularization function. If the loss function ℓ is convex, which it normally is, the problem is biconvex in I and P. Therefore, if we knew the depth map, we could find the all-in-focus image by solving a single convex problem, and vice versa.

The objective is composed of three parts: a loss between the reconstructed images and the true measured images, a regularization function applied to the all-in-focus image, and a regularization function applied to the depth distribution. We give some examples of ℓ, A_I, A_P, r_I, and r_P below.

Reconstruction loss. Some canonical examples for the reconstruction loss include the sum of squares loss and the robust Huber loss [10].

Image Regularization. A canonical example for image regularization is (isotropic) total variation. To add total variation regularization, we let

$$A_I = (\nabla_x I, \nabla_y I),$$

where $\nabla_x I$ is the convolution of I with $(1, -1)$ and $\nabla_y I$ is the convolution of I with $(1, -1)^T$, and

$$r_I(\tilde{I}_1, \tilde{I}_2) = \sum_{i,j} |(\tilde{I}_1)_{ij}| + |(\tilde{I}_2)_{ij}|.$$

Depth regularization. We use a similar linear operator for the depth regularization term:

$$A_P = (\nabla_x P, \nabla_y P).$$

For the regularization term, we can make the distribution smooth spatially by letting

$$r_P(\tilde{P}_1, \tilde{P}_2) = \sum_{i,j,k} (\tilde{P}_1)_{ijk}^2 + (\tilde{P}_2)_{ijk}^2,$$

or only change at a few elements by using total variation:

$$r_P(\tilde{P}_1, \tilde{P}_2) = \sum_{i,j,k} |(\tilde{P}_1)_{ijk}| + |(\tilde{P}_2)_{ijk}|.$$

3.3. Proposed algorithms

Gradient descent. If all of the functions involved are smooth, we can apply gradient descent directly to problem (1). Even if the regularization functions are not smooth, we can still apply gradient descent using subgradients of the regularization functions. Gradient descent is often preferred to more complicated methods, such as the ADMM method described below, due to how easy it is to implement in frameworks like PyTorch.

Alternating minimization. As we described before, if the loss function ℓ is convex, problem (1) is a biconvex optimization problem. A promising algorithm for these classes of problems is alternating minimization, where, at each step, we fix I and minimize over P, and then fix P and minimize over I. Each step in this algorithm requires solving two convex optimization problems. We can warm start these optimization problems to make them converge faster.

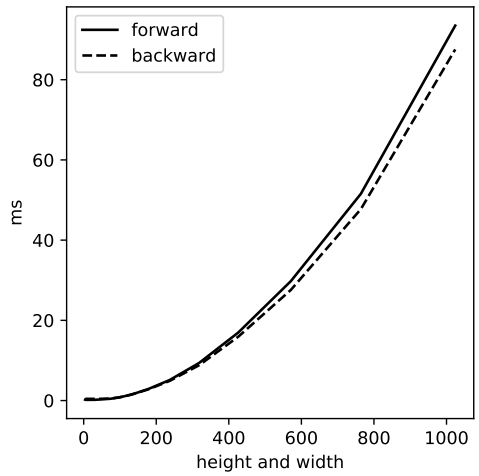


Figure 1: Timing of forward and backward image formation model.

ADMM. We can reformulate problem (1) as

$$\begin{array}{ll} \text{minimize} & L(I,P) + r_I(\tilde{I}) + r_P(\tilde{P}) \\ \text{subject to} & A_I I = \tilde{I}, \quad A_P P = \tilde{I}, \end{array} \tag{2}$$

with additional variables $\tilde{I}$ and $\tilde{P}$ and apply ADMM [1] with the splitting (I, P) and $(\tilde{I}, \tilde{P})$. The (I, P) update requires solving a (differentiable) nonconvex optimization problem, which we can do using gradient descent.

3.4. Implementation

We implemented the three aforementioned algorithms in PyTorch [18], a Python package for numerical computation and automatic differentiation on multi-dimensional arrays that supports multiple computation architectures (*e.g.*, CPUs, GPUs, TPUs). By implementing the image formation model in PyTorch, it makes it easy to automatically compute the gradient of L with respect to L and P for a wide range of loss functions. In figure 1, we show the timing of the forward and backward pass of the image formation model on an unloaded Nvidia 1080 TI GPU for $N = 10$, $K = 5$, $\max_{i,j} C_i(d_j) = 11$, and for varying height and width.

4. Results

We evaluate our approach first on simulated images and then on images captured using an Android smartphone.

4.1. Simulated data

Image and focal stack. We generate an image with parameters $H = W = 64$, $f = 0.05$, $N = 2.4$, focused at $N = 6$ distances

$$S = (0.8, 1.0, 1.3, 1.7, 2.0, 2.2).$$

We add uniform noise between -0.005 and +0.005 to each image in the focal stack. The synthetic image, the depth, and the focal stack is visualized in figure 2.

We discretize depth into $K = 2$ bins,

$$d = (1, 2),$$

and consider a true depth map where the left half of the image is at depth 1 and the right half of the image is at depth 2.

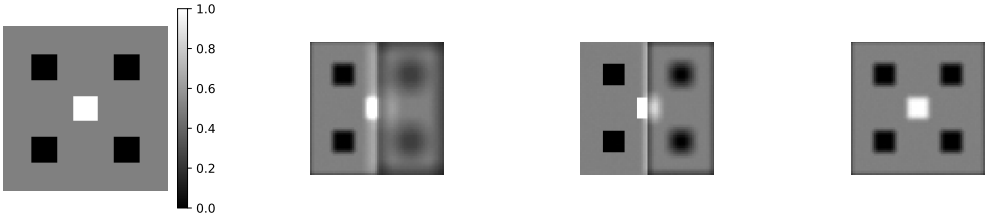
Results. We used TV regularization on the all-in-focus image (weighted by 0.1) and TV regularization on the probability distribution (also weighted by 0.1), and the sum of squares loss as the reconstruction loss, and ran gradient descent for 10000 iterations using a step size of 0.01. We initialized gradient descent with the mean of the focal stack, and initialized the depth distribution as uniform. The algorithm took 31.2 seconds to complete, so 3 milliseconds per iteration. In figure 3 we show the reconstructed all-in-focus image and depth map, *i.e.*, the final iterates of the algorithm. In figure 4, we plot the convergence of the algorithm, in terms of the objective, the PSNR with respect to the real image, and the root mean-squared-error (RMSE) of the depth. We find that the algorithm is able to roughly double the PSNR and bring the depth RMSE to zero. Qualitatively, we find that the reconstruction looks exactly like the original image.

4.2. Android images

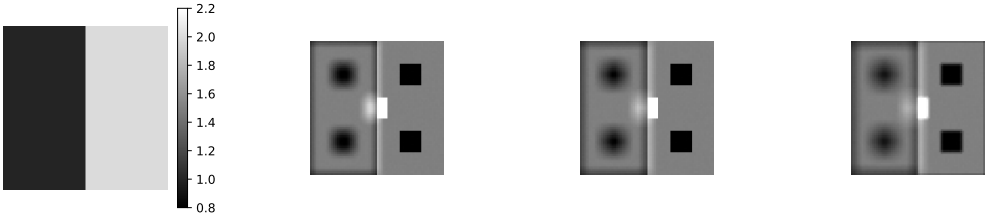
To capture our data to test our algorithm, we used the open source Open Camera app [7]. This allowed us to take consistent focal stacks with 100 different settings. We then sub-sampled the focal stacks to analyze specific focal distances.

Calibration. The focus distance of each image in the stack is unknown a priori, so we take a full focal stack of known test image (a grid of circles) at various distances to measure the empirical point spread function radius as a function of both distance to subject and camera focus; see figure 6. To measure the radius of the point spread function, we first identify all of the circles in the grid using the Hough Circle Transform [9]. Then we find the average thickness of the region at the edge of the circle that is between 10% and 90% of the observed intensity range; see figure 5. Assuming a disc shaped point spread function, the radius of the kernel is 0.72 times the thickness of this ring.

Equally as important, we used this calibration process to determine magnification as a function of focal distance (i.e.

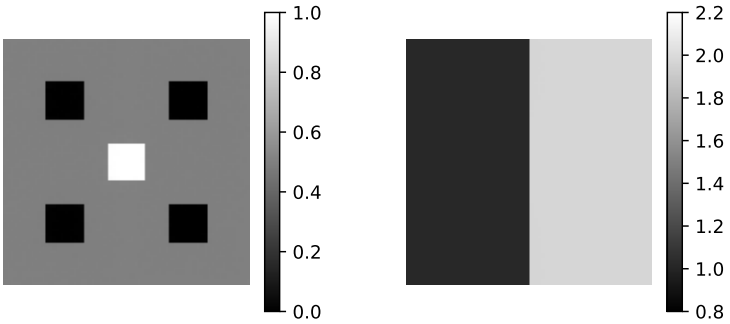


(a) True image. (b) $S = 0.8$. (c) $S = 1.0$. (d) $S = 1.3$.



(e) True depth. (f) $S = 1.7$. (g) $S = 2.0$. (h) $S = 2.2$.

Figure 2: Simulated focal stack.



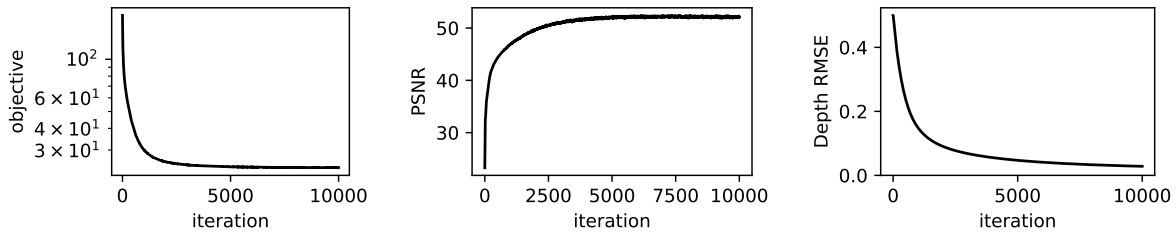
(a) Reconstructed all-in-focus image. (b) Reconstructed depth.

Figure 3: Reconstructions.

lens breathing); see figure 6. To determine the magnification of a blurring image, we found the center of each circle in the grid (a metric which is invariant to blur) and computed the total distance between all circles and their mutual center. This proved to be a reliable metric of the image magnification. We resize images by the inverse of their magnification to align them before we run the algorithm to determine the depth map.

Focal stack. We captured focal stacks of 100 images at focal distances ranging from 0.1 meters to infinity. All other parameters (ISO, aperture, white balance, etc) were held constant. The images were captured using a Samsung Galaxy S8 Active smartphone (f/1.7, $f \approx 0.00425$ meters).

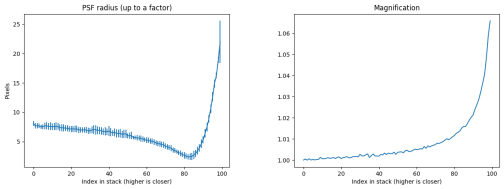
Results. We captured an image of three planar objects with abundant textured details at three depths: 10cm, 20cm, and 30cm. We then took the frames in our focal stack that



(a) Objective versus iteration. (b) PSNR versus iteration. (c) Depth MSE versus iteration.

Figure 4: Simulated data optimization results.

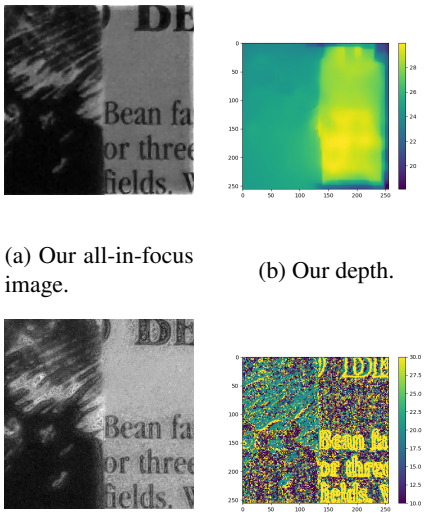
Figure 5: To calibrate the camera, we identify the circles in the test image and measure the thickness of the blurred border.



(a) The radius of the PSF for each frame, with the subject at 50cm. (b) The magnification of different frames in the focal stack.

Figure 6: Calibration data.

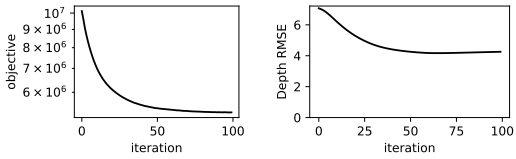
are in focus at 10cm, 20cm, and 30cm and use those to reconstruct a depth map and an all-in-focus image. We evaluate our method on a patch of the image directly on the boundary between the object at 20cm and 30cm. As a baseline, we implement an alternative depth estimation algorithm which for each pixel finds the focal depth which produces the highest gradient. In this setting, our algorithm



(a) Our all-in-focus image. (b) Our depth.

(c) Baseline all-in-focus image. (d) Baseline depth map.

Figure 7: Our model versus a maximum gradient baseline. The true depth is 30cm in the right half of the image, and 20cm in the left half.



(a) Objective versus iteration. (b) Depth MSE versus iteration.

Figure 8: Real world data optimization results.

estimates depth with an RSME 4.26cm, while the baseline achieves an RSME of 10.20cm. The baseline behaves poorly in areas where the sharp image has low gradients.

Our method behaves comparatively better because it uses total variation regularization to leverage information from the surrounding area. See figure 7 for the resulting all-in-focus image and depth map for our method and the baseline, and figure 8 for the optimization convergence plots.

5. Discussion

Our model performs well on synthetic data, achieving a near perfect reconstruction of both the true depth map and the true image. On real world data, we produce very sharp reconstructions of the all-in-focus image, but fairly noisy depth maps. This may be due to several approximations we make in our forward image formation model, *e.g.*, depth occlusions are computed slightly incorrectly. The diameter of the point spread function is measured empirically, and as such may have a certain degree of error. The model also takes a long time to run (several minutes for the 256x256 patch seen in figures). Further refinement is needed to simulate forward image formation accurately and quickly, but we believe that this framework provides a principled approach to solving depth from defocus as an inverse problem.

6. Conclusion

In this project we considered the problem of simultaneously extracting the all-in-focus image and the depth of each pixel in the image from a focal stack. We framed this problem as an inverse problem, with a biaffine image formation model, and proposed a number of solution algorithms. We applied gradient descent method to simulated images as well as images taken with a Samsung Galaxy S8 Active Android smartphone, and found the method to improve substantially on the maximum gradient baseline. Future work includes further investigation into efficient computational methods for this problem, more principled or automated methods for determining the camera parameters, and finally, an implementation of these methods in real-time on a smartphone using warm-starting.

References

[1] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[2] R. Brunner and T. Chen. Dynamic autofocus operations, 2014. US Patent 8,908,083.

[3] S. Chaudhuri and A. N. Rajagopalan. *Depth from defocus: A real aperture imaging approach*. Springer Science & Business Media, 2012.

[4] J. Ens and P. Lawrence. An investigation of methods for determining depth from focus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(2):97–108, 1993.

[5] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017.

[6] P. Grossmann. Depth from focus. *Pattern Recognition Letters*, 5(1):63–69, 1987.

[7] M. Harman. Open Camera Android App. `http://opencamera.org.uk/`.

[8] B. Horn. Focusing. *Project Mac Artificial Intelligence Memo*, 160, 1968.

[9] P. Hough. Machine analysis of bubble chamber pictures. In *Conf. Proc.*, volume 590914, pages 554–558, 1959.

[10] P. Huber. Robust estimation of a location parameter. In *Breakthroughs in Statistics*, pages 492–518. Springer, 1964.

[11] A. Ishikawa and T. Nihoshi. Autofocus system and microscope, 2006. US Patent 7,071,451.

[12] R. Jarvis. Focus optimisation criteria for computer image processing. 1976.

[13] R. Jarvis. A perspective on range finding techniques for computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2):122–139, 1983.

[14] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)*, pages 239–248. IEEE, 2016.

[15] F. Mannan and M. S. Langer. What is a good model for depth from defocus? In *2016 13th Conference on Computer and Robot Vision (CRV)*, pages 273–280. IEEE, 2016.

[16] S. Nayar and Y. Nakagawa. Shape from focus. *IEEE Transactions on Pattern analysis and machine intelligence*, 16(8):824–831, 1994.

[17] I. Ohnuki, A. Akashi, T. Kadohara, and M. Higashihara. Autofocus device, 1990. US Patent 4,969,003.

[18] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. 2017.

[19] A. Pentland. A new sense for depth of field. *IEEE transactions on pattern analysis and machine intelligence*, (4):523–531, 1987.

[20] A. Roy and S. Todorovic. Monocular depth estimation using neural regression forest. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5506–5514, 2016.

[21] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages I–I. IEEE, 2003.

[22] L. Shih. Autofocus survey: a comparison of algorithms. In *Digital Photography III*, volume 6502, page 65020B. International Society for Optics and Photonics, 2007.

[23] M. Subbarao and G. Surya. Depth from defocus: A spatial domain approach. *International Journal of Computer Vision*, 13(3):271–294, 1994.

[24] S. Suwajanakorn. Depth from focus with your mobile phone. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[25] H. Tang, S. Cohen, B. Price, S. Schiller, and K. N. Kutulakos. Depth from defocus in the wild. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 2740–2748, 2017.

[26] Y.-K. Yoon, Y.-g. Lee, and M.-W. Kim. Autofocus method for a camera, 2008. US Patent App. 12/037,153.

[27] S. Zhuo and T. Sim. Defocus map estimation from a single image. *Pattern Recognition*, 44(9):1852–1858, 2011.