Coded Aperture Ranging

Michael Wang

Submitted to Sibley School of Mechanical and Aerospace Engineering

in partial fulfillment of the requirements for the degree of

Master of Engineering in Aerospace Engineering

at

CORNELL UNIVERSITY

Advisor: Professor Dmitry Savransky

May 24, 2018

Contents

Acknowledgments

This project would not be possible without the assistance and support of Professor Dmitry Savransky of Sibley School of Mechanical and Aerospace Engineering at Cornell University. His technical expertise and guidance are essential to the completion and success of the project. I would also like to thank Engineering Learning Initiatives (ELI) for funding the project.

Abstract

Coded Aperture Ranging (CAR) is an imaging technique that can extract both depth information and an all-focused image from a single captured image by making a minor modification to a conventional camera system. Unlike other camera systems that require additional apparatus for depth perception, CAR systems require only a single camera and a partially masked aperture. The masked aperture enables the generation of a depth map just from a single image via Depth from Defocus techniques. The depth map can then be used to create an all-focused image from the original blurred image. This technique has applications in systems where size and weight constraints are paramount and image depth map generation is necessary. The goal of this project is to implement and demonstrate CAR on a conventional DSLR and explore its advantages and shortcomings.

Background

Coded aperture is originally used in astronomy for X- and gamma ray imaging systems. Since these high-energy waves cannot be focused with conventional lenses, coded apertures are utilized to distort the incoming light. The captured data are then reconstructed to obtain the properties of the source. This methodology is then adopted in computational photography, in which the source is visible light and the instrument is any high resolution imaging system, such as a DSLR.

We can model image formation as a convolution with additive noise:

$$y = f_k * x + \eta \tag{1}$$

where y is the captured image, f_k is the scaled aperture geometry, x is the original sharp image, η is noise, and $*$ is the convolution operator. For conventional cameras, f_k is close to circular. However, for coded aperture ranging, f_k can take on any geometry. It turns out the choice of f_k affects the robustness of deconvolution (inverting f_k to obtain x), which in turn affects the quality of depth estimation and generation of an all-focused image.

Depth from Defocus

The main effect that is leveraged in coded aperture ranging is depth from defocus:

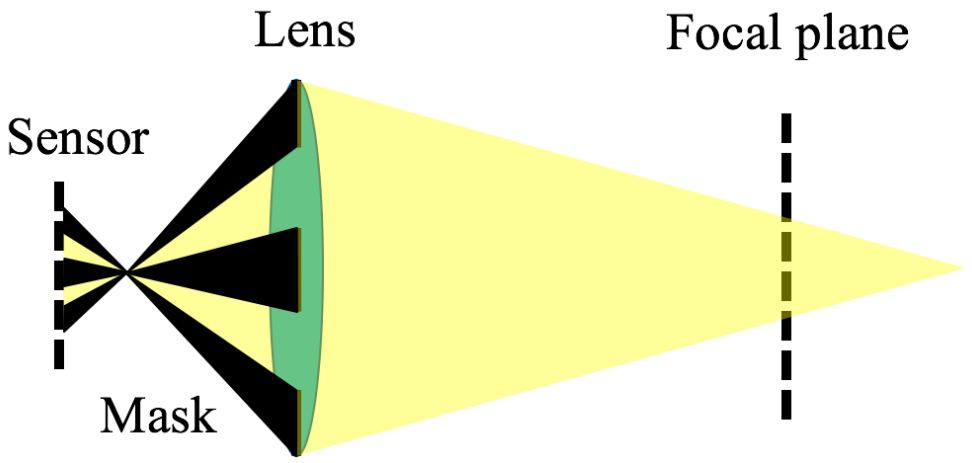


Figure 1: Depth from Defocus

If an object is directly on the focal plane, the image will appear on the sensor plane, resulting in a sharp image. If an object moves away from the focal plane in either direction, the image will be a blurred version of the object, creating a "circle of confusion". Theoretically,

if the amount of blur is known perfectly and deblurring the image with different kernels are sufficiently unique, one can determine the distance of the object relative to the focal plane. Since we know the focal plane distance, we can thus calculate depth. However, in practice, this is difficult to accomplish with conventional circular apertures. This is due to the distribution of zero frequencies of the Fourier transforms of the aperture at different scales.

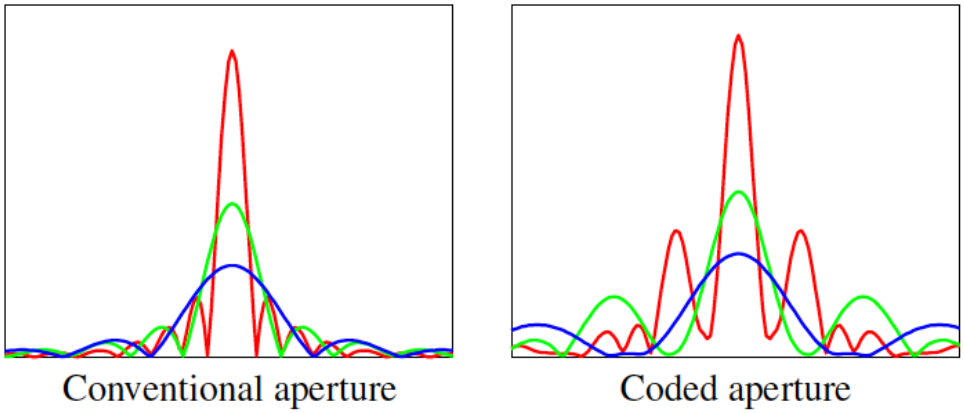


Figure 2: (left) Conventional circular aperture; (right) Coded Aperture Fourier Transforms [1]

As shown in Figure 2, the distribution of zero frequencies for coded aperture vary much more across scales than conventional circular aperture [1]. Since convolution in spatial domain is multiplication in frequency domain, from Equation 1 we can see that x is more likely to be sufficiently unique after deconvolution with different scales for coded aperture. This demonstrates the power of coded aperture. While coded aperture permits less light onto the optical sensor, depth discrimination is more easily achieved.

Range vs. Resolution

There is an inherent tradeoff between the range of depths for a certain focal plane lock and the resolution of the depth estimation. First, we have the lens equation

$$\frac{1}{s} + \frac{1}{f_p} = \frac{1}{f} \tag{2}$$

where s is the distance between the optical sensor and the optical center, f_p is the focal plane distance relative to the optical center, and f is the focal length of the lens. Using the same relation, we can write:

$$\frac{1}{v} + \frac{1}{o_d} = \frac{1}{f} \tag{3}$$

where v is the image distance from optical center and o_d is the object distance from optical center. Unless $o_d = f_p$, the image will not lie exactly on the sensor. Therefore, we can calculate the diameter of the Point Spread Function (PSF), D_{PSF}, which is equivalent to the size of the blur kernel, using similar triangles:

$$D_{PSF} = \frac{D_{ap}|s - v|}{v} \tag{4}$$

where D_{ap} is the diameter of the aperture, which would equal the diameter of the coded mask geometry. Using Equations 2, 3, and 4, we get:

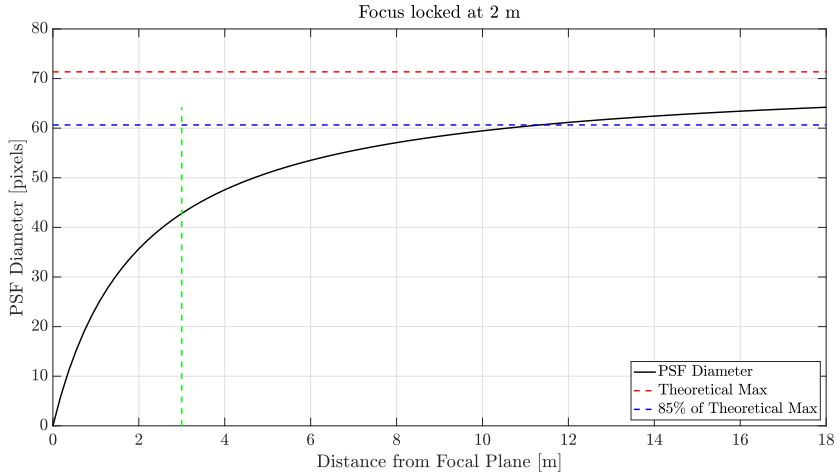


Figure 3: Diameter of PSF

As you can see in Figure 3, the diameter of the PSF approaches a certain maximum value as the distance from the focal plane approaches infinity. This is due to the fact that light rays from an object at infinity will appear parallel. The slope of this plot is directly proportional to the expected resolution of the CAR system. This is because a larger range of PSF diameters result in greater disparity between scaled kernels for a given variation in depth.

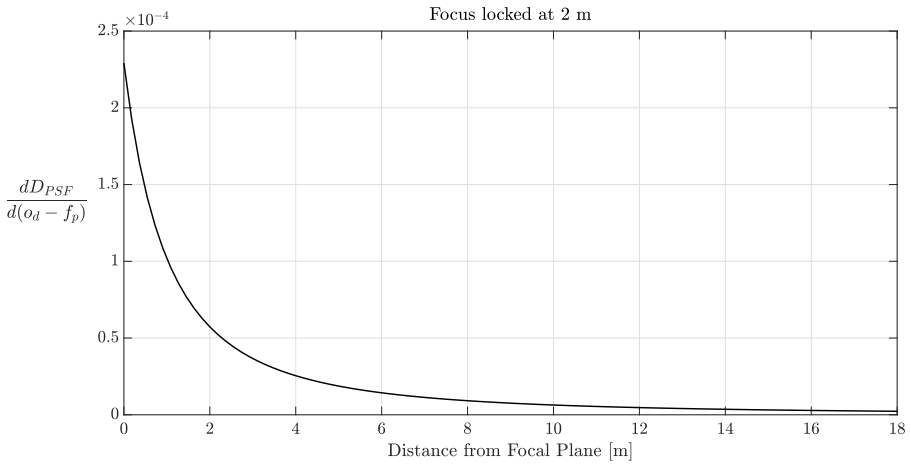


Figure 4: Slope of Figure 3

As seen from Figure 4, the turning point (point of diminishing return) occurs at around 2-4 meters from the focal plane. The same analysis can be done for a sweep in focal plane distance:

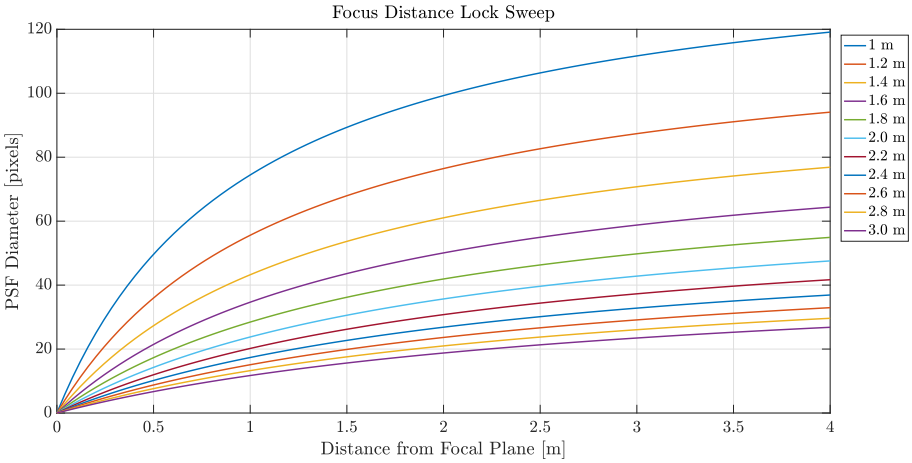


Figure 5: Sweep of Focal Plane Distance

As seen in Figure 5, as the focal plane distance decreases, the resolution increases (for small object distances), and vice versa. While this analytical formulation gives insight in the resolution of CAR, in practice, extremely small or large blur kernels prove to be challenging to work with due to sensor resolution limitations and deconvolution error respectively, which can be seen later.

Literature Review

Most of the work in this project is based on *Image and Depth from a Conventional Camera with a Coded Aperture* by Levin et. al. [1]. As shown by Levin et. al., defocus or blur in an image can be used to infer depth from the focal plane. This effect is known as depth from defocus (DFD), and can be easily seen in cameras with conventional apertures. However, in practice, it is very difficult to distinguish depth in images from conventional apertures, as shown earlier. To alleviate this problem, Levin et. al. introduced a coded mask that is able to amplify reconstruction errors when an image is deconvolved with a smaller kernel. This is due to the fact that the locations of the zero frequencies of the mask's Fourier transforms vary much more greatly with scale for coded aperture than for a conventional aperture. Levin et. al. utilizes the Kullback-Leibler divergence metric that measures the robustness of a particular filter to come up with an optimal mask design, which will be utilized by this project. Additionally, Levin et. al. introduced two blind deconvolution algorithms, namely deconvolution with Gaussian priors and deconvolution with sparse priors, which will be utilized to calculate reconstruction errors [1].

The blur kernel or PSF is estimated by finding a 2D mask that best explains the blurring between two images. Yang et. al. uses a simple pseudoinverse with least squares that minimizes the Frobenius norm of the estimated blur kernel minus the expected kernel under no noise [2]. Joshi et. al. solves the least squares solution that minimizes the reconstruction error plus a regularization term that is the norm of the gradient of the kernel [3]. Mannan et. al. solves the same problem as Joshi et. al. but includes an additional constraint for unit sum of elements of the mask.

Levin et. al. utilizes straightforward reconstruction error for depth estimation [1]. In contrast, Wang et. al. introduced a Structured Similarity index (SSIM) that can be used to compare the similarity of two image patches [4].

There are other interesting implementations of coded aperture. Zhou et. al. utilizes two complementary coded aperture patterns to conduct depth estimation and generate all-focused image [5]. Sellent et. al. combine depth estimation from coded aperture and optical flow to improve velocity estimation relative to the focal plane [6].

Methodology

Blur Calibration

To calibrate the blur kernel for various depths, we must first take a focused picture of the calibration pattern at the desired focal plane distance. Due to the shape of the aperture, the camera auto-focus will not be reliable. Therefore, it must be manually focused. After the focused image is taken, the camera will be moved back in pre-determined increments while the focus is *locked*. This process will generate intentionally blurred images with the amount of blur related to a function of depth. At the very last increment, the calibration pattern will be replaced with a test scene. The scene is captured still with the locked focus.

PSF Estimation

After the calibration images are obtained, we now need to estimate the blur kernel for each calibrated depth. We can first convert Equation 1 into matrix form:

$$y = Xf_k + \eta \tag{5}$$

where X is a matrix and y, f_k, and η are vectors of their corresponding frequency domain components. X represents the focused image while y represents the blurred image. The matrix X has a block Toeplitz structure that allows the convolution to work out [14]. The optimal solution for Equation 5 is a least squares solution that minimizes the reconstruction error. Therefore, the minimization looks like:

$$\begin{aligned} f_k^* = \arg\min_{f_k} \ & \lambda_1 \|y - Xf_k\|_2 + \lambda_2 \|\nabla f_k\|_2 \\ s.t. \quad & \mathbf{0} \leq f_k \leq \mathbf{1} \\ & sum(f_k) = 1 \end{aligned} \tag{6}$$

The L2-norm of the gradient of f_k is added in the objective function to bias the solution towards a smooth blur kernel. This technique is known as Tikhonov regularization and is often used in least squares problems. In practice, $\lambda_2 \|\nabla f_k\|_2$ is much smaller than $\lambda_1 \|y - Xf_k\|_2$. Constraints include elements of f_k need to be bounded by 0 and 1 (since we are working with image double data) and the sum of the elements of f_k must equal 1 (for energy conservation). One way to implement this is to use MATLAB's `lsqnonneg()`. The first half of the constraint will be automatically satisfied with non-negative least squares. To enforce

the other constraints, we can augment the matrix X and vector y to include

$$\begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} f_k = 1$$

The other option is to use MATLAB's `quadprog()`. The constraints can be specified in the implementation of the program. If the second constraint is not exactly satisfied, one can normalize the output kernel. Through experience, `quadprog()` tends to run faster than `lsqnonneg()` with similar results.

Deconvolution

After the blur kernels are generated, we now need to deconvolve the test scene image with each of the blur kernels. One deconvolution algorithm is deconvolution with Gaussian priors [1]

$$\tilde{x}^* = \arg\min_x \lambda_1 \|F_k x - y\|^2 + \lambda_2 \|G_x x\|^2 + \lambda_3 \|G_y x\|^2 \tag{7}$$

where x is now a vector and F_k is now a block Toeplitz matrix. Again there are additional regularization terms where G_x and G_y are the matrix equivalents of convolving with Gaussian derivatives. This is formulated with the assumption that the image is a Gaussian distribution, which tends to over-smooth the result. The other option is deconvolution with sparse priors [1]

$$\begin{gathered} \tilde{x}^* = \arg\min_x \lambda_1 \|F_k x - y\|^2 + \sum_{i,j} \rho(x(i,j) - x(i+1,j)) + \rho(x(i,j) - x(i,j+1)) \\ \rho(z) = \|z\|^{0.8} \end{gathered} \tag{8}$$

This minimization prefers solutions with concentrated gradients over spread-out gradients. This is due to the nature of the heavy-tailed function $\rho(z) = \|z\|^{0.8}$, which prefers small number of large gradients over large number of small gradients. This means the resultant image will be sharper. While this is a much harder optimization problem and takes a lot longer to solve, sparse priors is preferred over gaussian priors.

Depth Determination

After we have obtained a set of deconvolved images, we move on to depth determination. We use a sliding window, as below:

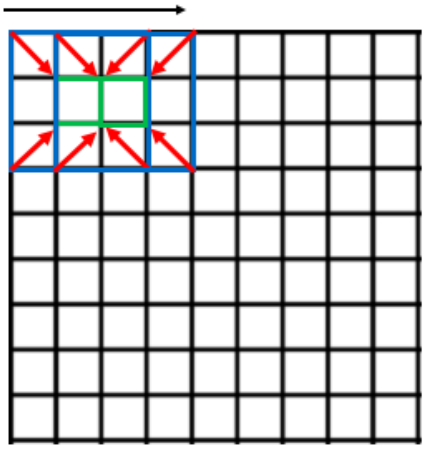


Figure 6: Sliding Window

to calculate the reconstruction error

$$r_k = \|y - f_k * \tilde{x}_k\|_2 \qquad k = 1, \ldots, n \tag{9}$$

where n is the number of blur kernels. Note that each blur kernel corresponds to a calibrated depth value. Therefore, we generate a depth map by calculating the reconstruction error of each local, sliding window using Frobenius norm, find the minimum reconstruction error and its associated kernel, and finally assign the calibrated depth value.
Another choice is to use the Structured Similarity Index (SSIM) [4]:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma xy + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{10}$$

where μ_x is the mean luminance of image x, σ_x is the standard deviation of x, and σ_{xy} is the covariance between x and y. The optimal depth value is found by the local window with the highest SSIM.

Weightings and Biases

We can train a set of weightings and biases on random image patches to minimize the depth estimation error. The optimization problem looks like:

$$\begin{aligned} \omega^*, b^* &= \arg\min_{\omega, b} \ \|f\left(\min\left(\omega_1 r_1 + b_1, \cdots, \omega_n r_n + b_n\right)\right) - d\|_2^2 \\ \omega &= \Big[\omega_1, \cdots, \omega_n\Big]^T \\ b &= \Big[b_1, \cdots, b_n\Big]^T \end{aligned} \tag{11}$$

where ω is a set of weights, b is a set of biases, f is a function that maps from the minimum residual to its calibrated depth, and d is a vector of correct depth values. The process in the previous subsection is repeated for each random image patch to generate m sets of residuals and corresponding correct depth value d. A helper function then takes these as inputs and calculates the objective function from Equation 11. A nonlinear search minimizer, such as MATLAB's `fminsearch()`, or nonlinear least squares solver, such as MATLAB's `lsqnonlin()`, is then used to minimize the objective function using the helper function.

All-focused Image

After an appropriate depth map is generated, the all-focused image can be easily constructed. To start, each color channel of the captured scene is deconvolved with each calibrated blur kernel. Temporarily combine the deconvolved channels using MATLAB's `rgb2gray()`. Generate a depth map using the methods mentioned earlier. For each value in the depth map, concatenate the corresponding RGB values from the deconvolved channels to form a color, all-focused image.

Experimental Setup

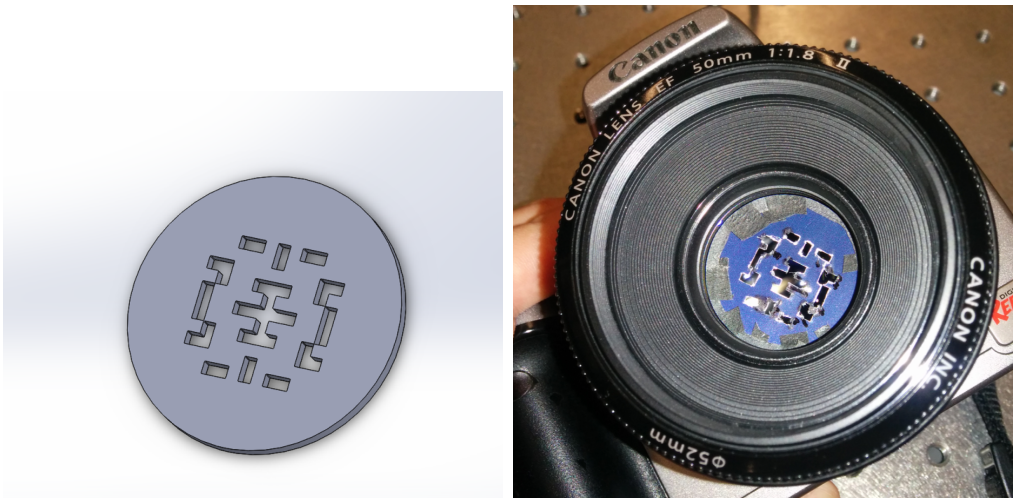
The list of equipment needed:

1. Camera (Canon Rebel XT in current set up) and tripod
2. Aperture mask
3. Lens (Canon EF 50mm f/1.8 II Camera Lens in current setup)
4. Calibration pattern
5. Test objects

Note that the lens is the old/discontinued Canon lens, not the newer STM version. This is crucial since the newer STM version cannot be taken apart without breaking the lens.

The procedure is as follows:

1. Create an aperture mask by cutting the stencil out of cardboard. Other options include 3D printing or printing on transparencies.



(a) Solidworks Model (b) Actual

Figure 7: Aperture Mask

   The aperture pattern is drawn on a 13x13 mm grid with 1x1mm squares. Dark tape is used to create a light seal around the rim of the aperture mask.

2. Follow the very detailed guide by Yosuke Bando to replace/add aperture mask to the lens [16].

3. Attach the calibration pattern on a flat surface such as the wall:

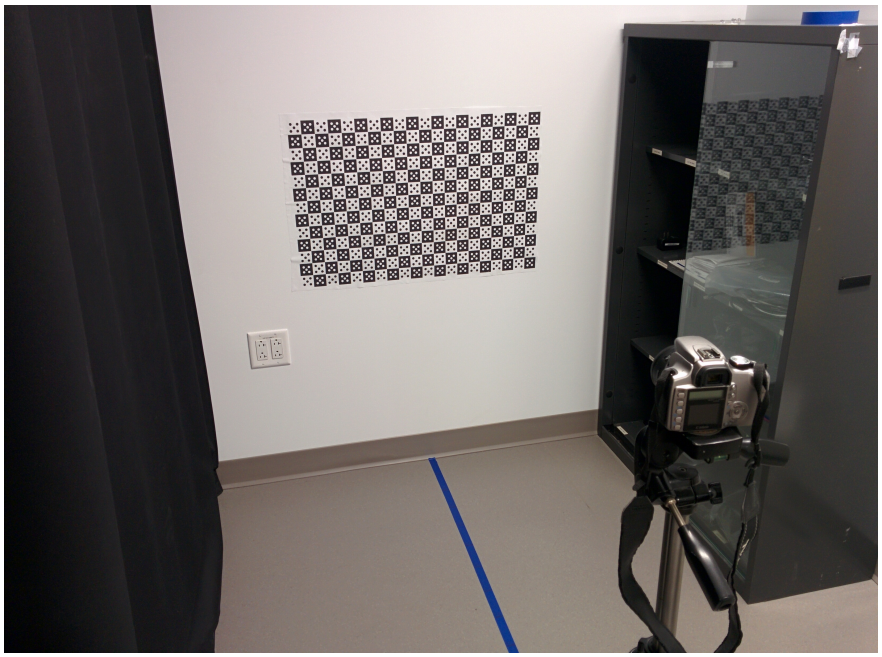


Figure 8: Experiment Setup

4. Set the tripod at the desired focal plane distance. Make sure camera is in full manual mode and manual focus. Manually focus the camera until the calibration pattern appears sharp. Set image capture format as RAW and JPG (for some reason RAW only mode still outputs undersampled image). Adjust shutter speed until standard exposure is reached. Set a timer on the camera so that your hand vibrations do not ruin the image.

5. Repeat this process for all desired calibrated depths with the focus fixed. Adjust the shutter speed regularly to maintain the same level of exposure.

6. After the last calibration image is taken, do not move the camera yet. Set up the test scene right in front of the calibration pattern. Adjust camera height so that test scene is centered in the image. Capture the test scene with the fixed focus.

7. The calibration pictures need to be aligned with each other. I have made a MATLAB GUI program, `edit_patches.m`, that can be used to align calibration images as well as image patches for training weights and biases.

8. Resize all calibration patterns to the size of the smallest pattern. Obtain blur kernels using least squares.

9. Deconvolve the captured scene with each of the blur kernels using deconvolution with sparse priors.

10. Use local, sliding window to calculate reconstruction error. Find the minimum error and its corresponding depth value to generate a complete depth map.

11. Utilize the depth map to generate an all-focused image. Of course, there would need to be deconvolved color channels.

Results

Major results will be displayed here.

$f_p = 2\ m,\ \text{RANGE} = 2.1 - 3.0\ m$

In the first example, the focus plane is locked at 2 m from the optical center. The blur kernels are:

Figure 9: Blur Kernels for 2m focus plane with range 2.1-3m

From left to right, top to bottom, the blur kernels correspond to depths of 2.1 to 3 m in 0.1 m increments. As expected, the blur kernels resemble the scaled geometry of the aperture mask.

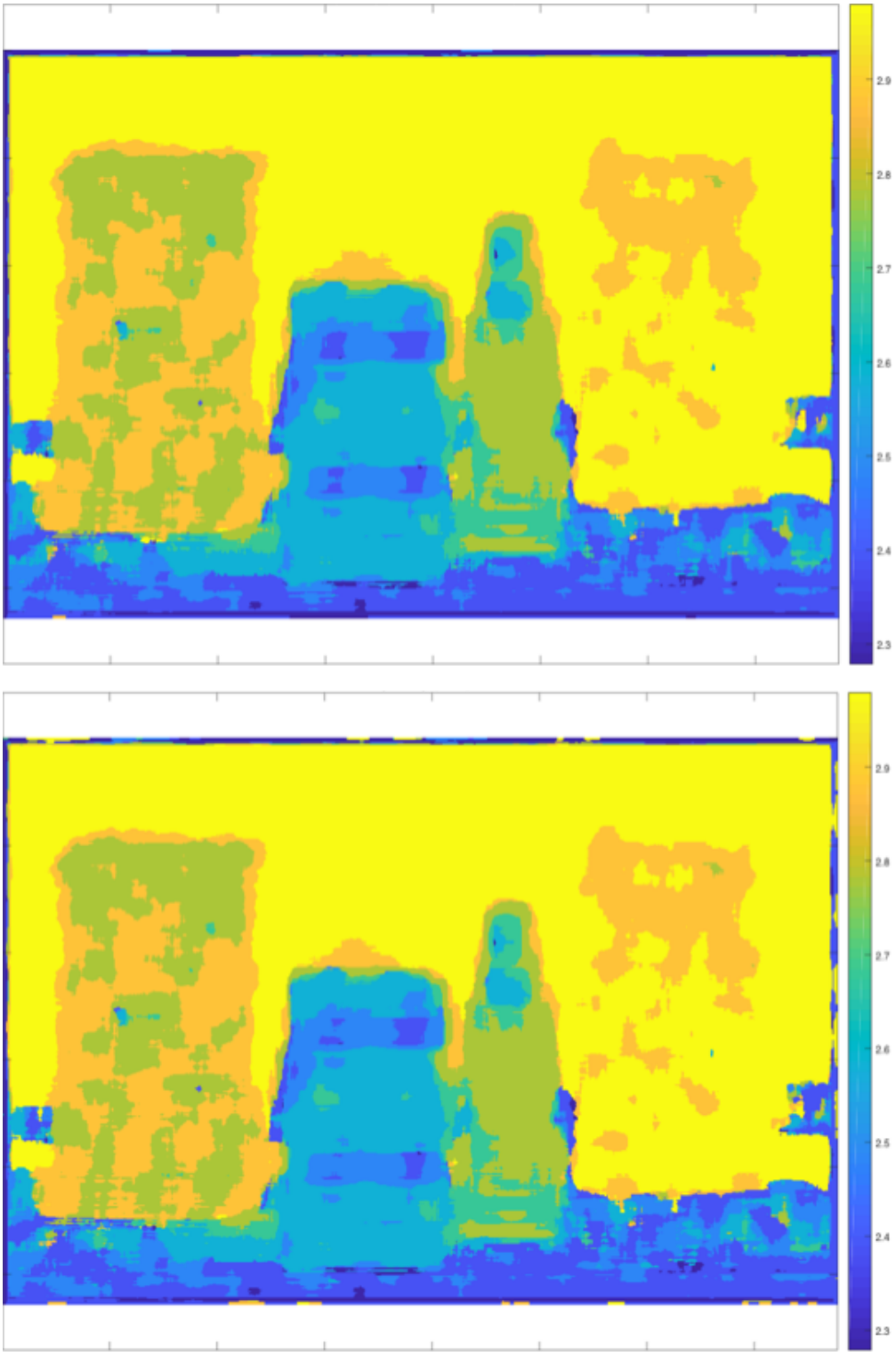


Figure 10: Depth Map for 2m focus plane. (Top) Reconstruction Error. (Bottom) SSIM

The units are in meters. As shown in Figure 10, the depth maps were able to distinguish among the objects and the background. Furthermore, besides the object on the right, the

depth maps accurately capture the depths of the objects relative to the camera. Furthermore, the background (wall) also has a correct depth value of 3 m. In addition, the depth map using reconstruction error appears almost the same as the depth map using SSIM. The differences are hard to spot, but they are there. This demonstrates both metrics are equivalent and interchangeable.

Figure 11: Scene Images. (Top) Original (Bottom) All-focused

Each object is 20 cm apart from its adjacent object. As shown in Figure 11, with the help of the depth map, the image was able to be decently focused. However, there are slight ringing effects on top of the lens boxes. In addition, the color is slightly red-biased. Future work can include windowed or segmented deconvolution in order to mitigate the ringing artifacts in the deconvolved image.

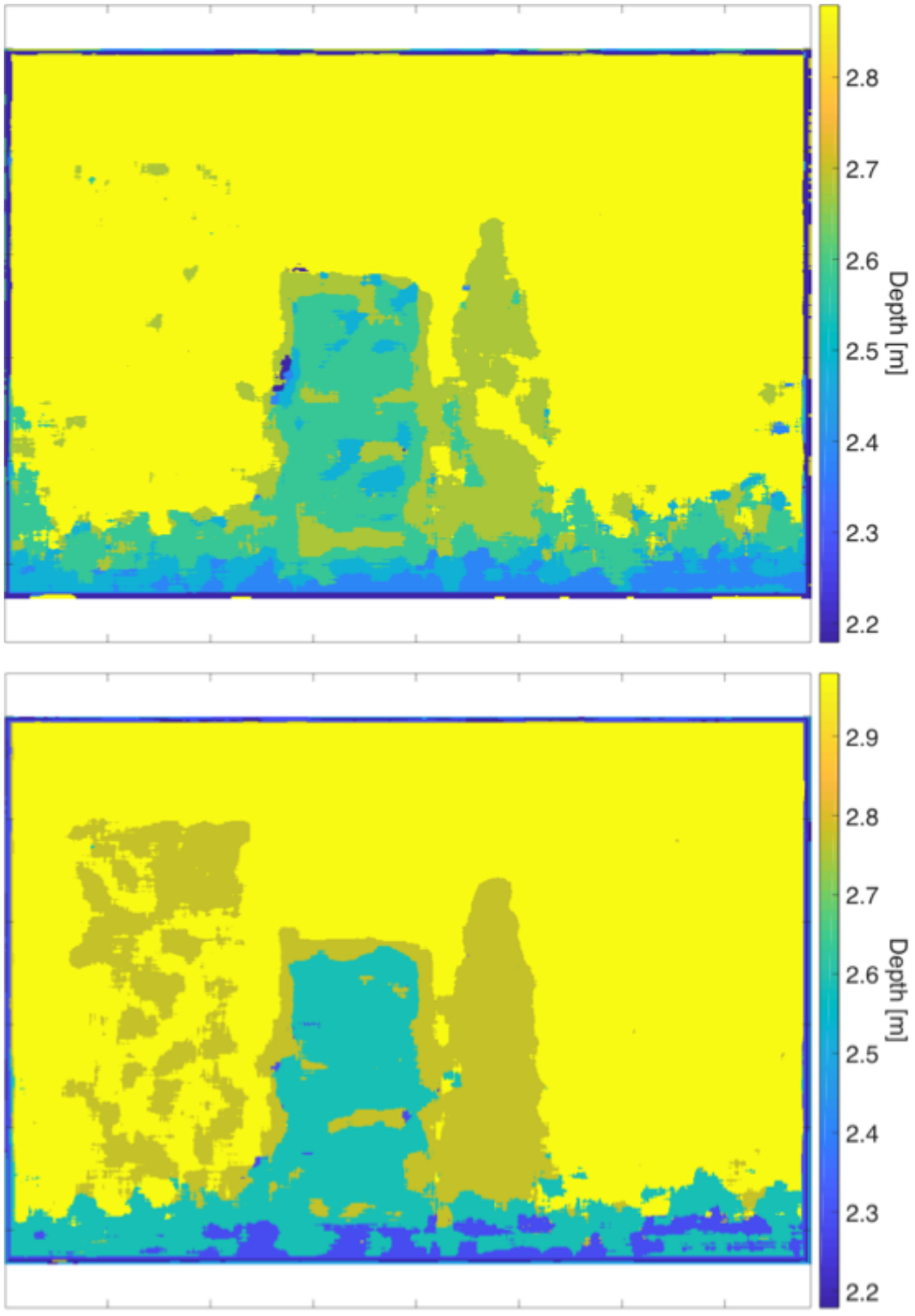


Figure 12: (Top) Pure weightings (Bottom) Weightings and biases

As shown in Figure 12, the depth maps with weightings and biases unfortunately do not perform better than the raw depth map in Figure 10. One possibility is that the sample of patches is too small. Future work can include implementing machine learning techniques to solve for weightings and biases.

$f_p = 1.3\ m$, RANGE $= 1.4 - 2.3\ m$

In the second example, the focus plane is locked at 1.3 m from the optical center. The blur kernels are:

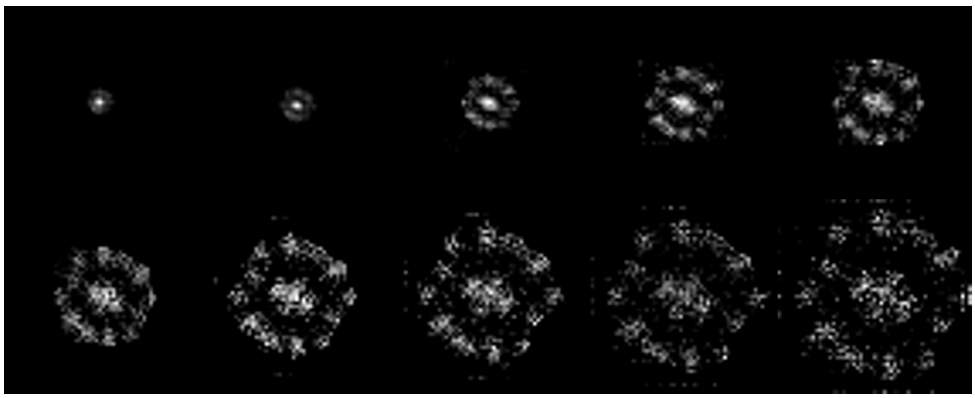


Figure 13: Blur Kernels for 1.3m focus plane with range 1.4-2.3m

From left to right, top to bottom, the blur kernels correspond to depths of 1.4 to 2.3 m in 0.1 m increments. The kernels increase in size much more rapidly over the same range due to the shorter focal plane distance, as shown in Figure 5. The last two blur kernels appear sparse. One possibility is that larger kernels suffer from diminished intensity due to its larger area. However, after increasing the camera exposure and environmental lighting, the blur kernels remain roughly the same.

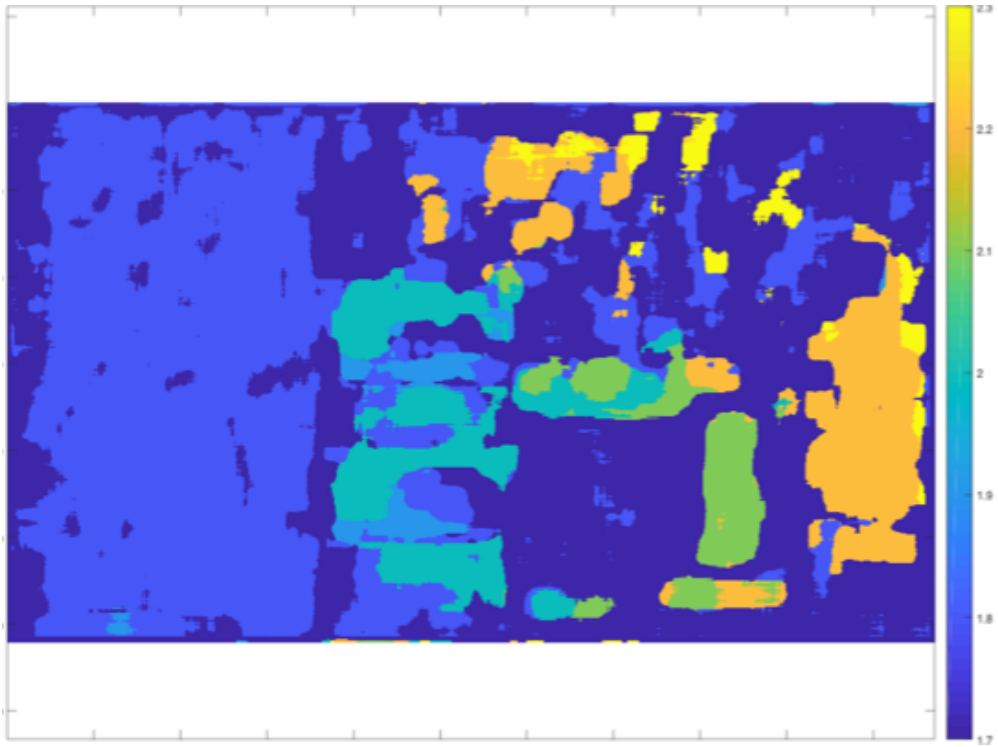


Figure 14: Depth Map for 1.3 m focus plane

The units are in meters. As shown in Figure 14, the depth map does not capture the correct depth for the background. However, the individual objects appear to be at the correct depths. The body of the red cup in the middle does not appear in the depth map. This is likely because the cup is smooth and without gradients except for the top edge. This highlights one disadvantage of CAR, in which featureless objects are difficult to detect.

Figure 15: 1.3 m focus plane. (Top) Original (Bottom) All-focused

Each object is 10 cm apart from its adjacent object. As shown in Figure 15, the quality of the all-focused image is greatly degraded due to a poor depth map.

Conclusion and Future Work

Given the right aperture mask as well as deconvolution algorithm, one can perform depth estimation and generate an all-focused image from a monocular camera modified with coded aperture. Furthermore, the depth estimation step takes only a few seconds on a modern CPU since the majority of the work is off-loaded onto the calibration step, which is a fixed overhead. Shortcomings of the CAR include necessity of external lighting source and difficulty of depth estimation with textureless objects. Future work could include improving the depth resolution, experimenting with various focal plane distances, experimenting with separate PSFs for sections of the captured image, interpolating/extrapolating PSFs, exploring minimum graph cut techniques for image segmentation, building a railing system for more precise calibration, and conducting trade studies on applications in spacecraft imaging and navigation.

References

[1] A. Levin, R. Fergus, F. Durand, and W. T. Freeman, "Image and depth from a conventional camera with a coded aperture," *ACM Trans. Graph.*, vol. 26, p. 70, 2007.

[2] J. Yang, B. Jiang, J. Ma, Y. Sun, and M. Di, "Accurate point spread function (psf) estimation for coded aperture cameras," 2014.

[3] N. Joshi, R. Szeliski, and D. J. Kriegman, "Psf estimation using sharp edge prediction," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, June 2008.

[4] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, pp. 600–612, April 2004.

[5] C. Zhou, S. Lin, and S. K. Nayar, "Coded aperture pairs for depth from defocus and defocus deblurring," *International Journal of Computer Vision*, vol. 93, pp. 53–72, May 2011.

[6] A. Sellent and P. Favaro, "Coded aperture flow," vol. 8753, pp. 582–592, 09 2014.

[7] F. Mannan and M. S. Langer, "Blur calibration for depth from defocus," in *2016 13th Conference on Computer and Robot Vision (CRV)*, pp. 281–288, June 2016.

[8] R. Raskar, *Computational Photography: Epsilon to Coded Photography*, pp. 238–253. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[9] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, pp. 1222–1239, Nov. 2001.

[10] J. S. Lee, "Coded aperture ranging techniques," pp. 1–25, May 2017.

[11] M. Masoudifar and H. R. Pourreza, "Image and depth from a single defocused image using coded aperture photography," *CoRR*, vol. abs/1603.04046, 2016.

[12] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, "Understanding and evaluating blind deconvolution algorithms," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1964–1971, June 2009.

[13] P. Favaro and S. Soatto, *3-D Shape Estimation and Image Restoration: Exploiting Defocus and Motion Blur*. 01 2007.

[14] P. Hansen, J. Nagy, and D. O'Leary, *Deblurring Images*. Society for Industrial and Applied Mathematics, 2006.

[15] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*. San Diego, CA, USA: California Technical Publishing, 1997.

[16] Y. Bando, "How to disassemble the canon ef 50mm f/1.8 ii lens," pp. 1–21.

Appendix

MATLAB Code

All code and data are on `https://github.com/michaelwang2/Coded_Aperture`. There are 7 folders representing 7 sets of experimentations. Common to all 7 are some important functions, which I have separated into a folder called "Important_Functions". Here is the full list:

1. `calcKer_lsqnonneg(B, S, szKer, l1, l2)`: This function calculates the blur kernel using MATLAB's `lsqnonneg()` between a blurred image (B) and sharp image (S). szKer is the size of the kernel. l1 and l2 are parameters for the regularization terms.

2. `calcKer_quadprog(B, S, szKer, szFilt, l1, l2, l3, l4, l5)`: This function does the same thing as above but using MATLAB `quadprog()` instead.

3. `conv2D2mtx(szKer, mask)`: This function is a helper function for the above. It generates the matrix form of a convolution for a mask.

4. `cropKernel(mKer, pix)`: This function crops a kernel equally on all 4 sides. pix is the number of pixels to crop on each side.

5. `deconvL2_frequency(I,filt1,we)`: This function performs deconvolution using Gaussian priors in the frequency domain (using fft) [1]. we is the weighting factor for the Gaussian derivative regularization terms.

6. `deconvL2(I,filt1,we,max_it)`: This function performs deconvolution using Gaussian priors in the spatial domain using conjugate gradient method [1].

7. `deconvSps(I,filt1,we,max_it)`: This function performs deconvolution using sparse priors in the spatial domain using iterative re-weighted least squares [1].

8. `deconvL2_w(I,filt1,we,max_it,weight_x,weight_y,weight_xx,weight_yy,weight_xy)`: This is a helper function for the above [1].

9. `edit_patches.m`: This function is a MATLAB GUI script to re-orient, crop, and store calibration images or image patches.

10. `fftconv(I,filt,method)`: This function performs convolution using FFT [1].

11. `SSIM(O, B, K1, K2)`: This function calculates the Structured Similarity Index [4].

12. `weights.m`: This script uses training dataset to find a set of weightings and biases to improve depth misclassification error through nonlinear optimizers.

The following list is the 7 main sets of experimentations and some comments.

1. **Current**: Initial Experimentation. Tried lens distortion correction using OpenCV. Depth map was not usable. Calibration pattern squares were too big.

2. **Currentv2**: Tried optimizing for weightings for the first time. Started using RAW images (.CR2) instead of JPG. Calibration pattern squares are too small.

3. **Currentv3**: Introduced `calcKer_lsqnonneg()` and `calcKer_quadprog()`. Used medium sized calibration pattern squares. Depth map is decent but still not great. Continued experimenting with weightings.

4. **Currentv4**: Used `deconvSps()` for the first time. Started utilizing sliding window for depth estimation. The best set out of all 7 sets. Depth map and all-focused image were successfully generated. Experimented more with weightings and biases, however, depth map with weightings and biases are ultimately inferior to raw depth map.

5. **Currentv5**: Use conference room TV monitor for calibration. Resized calibration pattern according to distance of camera to screen (to account for perspective shrinking). Collected multiple image patches. Depth map was not usable, most likely due to blurred edges caused by the display. Likely not high enough resolution.

6. **Currentv6**: Experimented with 1.3 m focal plane. Received decent results on depth map, although the background was not fully captured.

7. **Currentv7**: Experimented with increased exposure time and external lighting. Obtained similar results as Currentv6.

Coded Aperture Ranging

Michael Wang, Dmitry Savransky
Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY

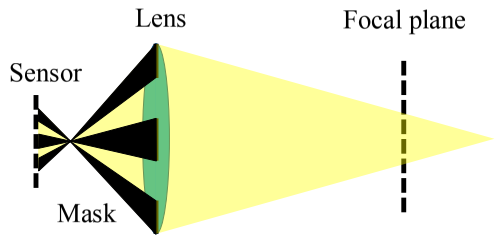
Cornell University

SIDS LABORATORY

Introduction

Coded Aperture Ranging (CAR) is an imaging technique that can extract both depth information and an all-focused image from a single captured image by making a minor modification to a conventional camera system. Unlike other camera systems that require additional apparatus for depth perception, CAR systems require only a single camera and a partially masked aperture. The masked aperture enables the generation of a depth map just from a single image via Depth from Defocus techniques. The depth map can then be used to create an all-focused image from the original blurred image. This technique has applications in systems where size and weight constraints are paramount and image depth map generation is necessary.

Depth from Defocus



- Blur in an image is created due to depth away from the focal plane.
- Amount of blur is a function of depth only.
- Coded Aperture amplifies depth discrimination over conventional aperture.

PSF Estimation

- Image formation is modeled as a convolution with additive noise:

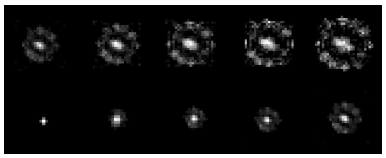
$$y = f_k * x + \eta$$

- Thus, finding the blur kernel can be done via a quadratic program that minimizes the reconstruction error as well as a regularization term:

$$f_k^* = \arg\min_{f_k} \lambda_1 \|y - f_k * x\| + \lambda_2 \|\nabla f_k\|$$

Blur Calibration and Depth Determination

- First, the focus of the camera is locked (at 2 m) and a focused image of the calibration target is taken.
- The camera then moves further from the calibration target at regular intervals to take blurred images of the calibration target.
- Pairs of blurred and focused images are used to estimate the blur kernel corresponding to the distance of the camera from the target, using the algorithm mentioned in the PSF estimation section.

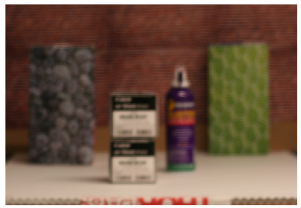


- As shown in the figure above, the estimated blur kernels are functions of the depth and the aperture shape. The larger kernels correspond to larger calibrated depths.
- The next step involves deconvolving the captured image with each of the blur kernels.
- Deconvolution with sparse priors is used to provide sharp images:

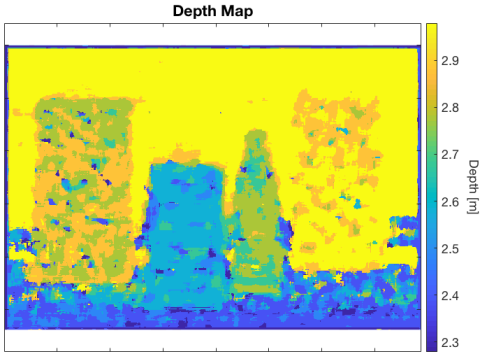
$$x^* = \arg\min_x |C_{f_k} x - y| + \textstyle\sum_{i,j} \rho(x(i,j) - x(i+1,j)) + \rho(x(i,j) - x(i,j+1))$$

$$\rho(z) = |z|^{0.8}$$

- Regions where the depth value matches the blur kernels' will appear sharp, and vice versa. Therefore depth for each local window is chosen based on the blur kernel that minimizes the reconstruction error after deconvolution.



Experimental Results



Raw depth map shows clear image segmentation between foreground and background as well as among objects. Resolution of ~20 cm is achieved.

Conclusions and future work

Given the right aperture mask as well as deconvolution algorithm, one can perform reliable depth estimation from monocular cameras with coded apertures. Furthermore, the depth estimation step takes only a few seconds on a modern CPU since the majority of the work is off-loaded onto the calibration step, which is a fixed overhead. Future work include demonstration of the Coded Aperture system on a smaller form factor such as a Raspberry Pi camera module. Also, further work on improving the depth resolution as well as trade studies on applications in spacecraft imaging will be conducted.

Acknowledgments and References

This research is funded by the Engineering Learning Initiatives.

[1] A. Levin, R. Fergus, F. Durand and W. Freeman, "Image and depth from a conventional camera with a coded aperture", *ACM Transactions on Graphics*, vol. 26, no. 99, p. 70, 2007.
[2] N. Joshi, R. Szeliski and D. Kriegman, "PSF Estimation using Sharp Edge Prediction", 2013.