Color-Filtered Aperture for Image Depth and Segmentation

Leron Julian
Carnegie Mellon University
Computational Photography Fall 2020 - Final Project
ljulian@andrew.cmu.edu

Abstract

From a single captured image, depth can be computed through the use of a red, green, and blue (RGB) colored aperture placed in the lens of a DSLR camera. Due to the misalignment of each plane, a different wavelength causes a shift disparity in each channel in which depth can be computed. This allows object separation between foreground and background through alpha matting for many post-capture image editing techniques.

1. Introduction

Various depth estimations techniques have been researched in the recent years for a variety of applications including 3D reconstruction and modeling, post-editing affects, and refocusing. However, many of these techniques involve using a multi-camera setup such as in the case of stereo vision and some lightfield-plenoptic cameras. Although effective, these camera setups are not practical for numerous reasons. For example, capturing a scene at different positions in which the object is non-moving may not always be obtainable. Also, plenoptic imaging systems can be clunky, large, and very expensive. The goal of this project is to develop an inexpensive, compact solution that can capture depth from a single capture. This project uses an RGB color filter placed inside of the aperture of a DSLR lens to compute depth from color misalignment. The position of the color filter [Figure 1] is oriented in such a way that an object that is farther from the focus depth will have a right-shift in the red channel, an up-shift in the green channel, and a left-shift in the blue channel. From this color misalignment, image depth from a single capture can be computed. This also enables segmentation and separation of foreground and background for an object in focus through alpha matte optimization.

Figure 1: Color-Filter aperture set-up and orientation in a DSLR Lens.

2. Previous Work

Coded Aperture Several research studies have experimented with placing various masks inside of the aperture of camera lenses with the purpose of deconvolution and depth from defocus such as in [3].

Color-Filtered Aperture The original idea of utilizing color-filtered apertures in imaging systems was proposed by [1]. This original method estimated disparities by using a squared intensity difference measure for high-pass filtered images. Although effective, their solution was not portable and cannot be utilized in modern day imaging systems to compute depth from misalignment in a single image capture.

3. Color-Filtered Aperture

In this project an RGB color filter is placed inside of the aperture of a DSLR lens to compute depth from color misalignment. The position of the color filter is oriented in such a way that an object that is farther from the focus depth will have a right-shift in the red channel, an up-shift in the green channel, and a left-shift in the blue channel. These shifts are ultimately due to geometric shift and not chromatic aberration. This is better demonstrated with a red and green 2-channel color filter example in [Figure 2]. The aperture is placed in such a way that its optical center aligns with the X and Y axes of the image sensor.

The color filter used for this project was an inexpensive color correction lighting gel that was cut into squares and

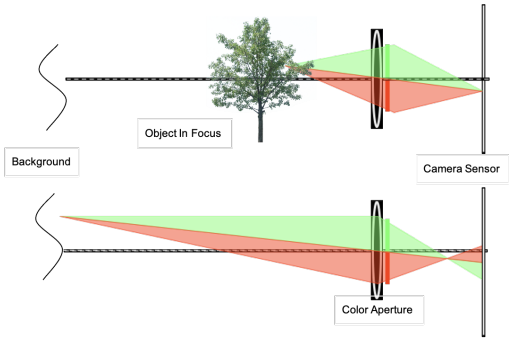


Figure 2: How colors get misaligned.

stuck together using a circular cutout from a cardboard box and placed in the orientation shown in [Figure 1]. The lens for this project used was a Canon EF 50mm $f/1.8$ lens that was disassembled in which the color-filtered aperture was placed on the aperture of the lens. The lens is attached to a Canon T5 body. [Figure 4] shows an image captured from the RGB color-filtered lens set up discussed in this section.

4. Depth Estimation

After capturing an image with the proposed set-up mentioned in section 3, image depth can now be computed by estimating the disparity between the shifted RGB planes. Let d be the hypothesized disparity at pixel (x, y) between the 3 RGB planes: I_r, I_g, and I_b. Therefore, due to the misalignment shifts mentioned in the previous section, we need to measure the quality of a match between: $I_r\left(x+d, y\right)$, $I_g\left(x, y-d\right)$, $I_b\left(x-d, y\right)$. These 3 values are represented at 3 different wavelengths and therefore it cannot be expected that these planes have similar intensities. However, proposed by [5] colors in un-shifted images captured by a regular camera form an elongated clusters when plotted in 3D space. As the disparity between the RGB channels increase, the clusters become less elongated and more isotropic. Therefore, by using a select number of pixels within a certain window of an image $w\left(x, y\right)$ belonging to one cluster, the magnitude of the cluster can be used as a correspondence measure. This can ultimately help solve for the true disparity d.

Further, with a hypothesized disparity d, consider a set of pixels within a certain window $w\left(x, y\right)$ of the image as: $S_I = \left\{\left(I_r\left(s+d, t\right), I_g\left(s, t-d\right), I_b\left(s-d, t\right)\right) | (s, t) \in w\left(x, y\right)\right\}$ Therefore, by minimizing the color alignment measure below, an optimal disparity d can be obtained:

$$L(x, y; d) = \frac{\lambda_0 \lambda_1 \lambda_2}{\sigma_r^2 \sigma_g^2 \sigma_b^2}$$

Where $\lambda_0 \lambda_1 \lambda_2$ are the eigenvalues of the $3x3$ covariance matrix Σ and $\sigma_r^2 \sigma_g^2 \sigma_b^2$ are the diagonal elements of Σ. Therefore, as L gets smaller, the clusters become more elongated and the RGB components are correlated meaning that the disparity d between the planes are smaller. Adversely, as L increases, so does d meaning that the RGB components are less correlated and the clusters become more isotropic.

4.1. Color Lines Model

Below in [Figure 3] is a visual of the color lines model from a sample image captured from a regular lens with a true disparity of $d = 0$. A small window of pixels (outlined by the red box) is taken as a sample cluster. Plotting the window in the RGB space with a true disparity of $d = 0$, shows that the cluster is more elongated with a low color alignment measure L. As d is increased, the cluster becomes more elongated and L increases as well meaning that the RGB planes are less correlated

Now, by using the color alignment measure, the disparity d that minimizes $L(x, y; d)$ at each pixel (x, y) can be used to ultimately find the optimal match between the RGB planes to obtain depth at each pixel. [Figure] shows a sample computed depth map after searching for an optimal d.

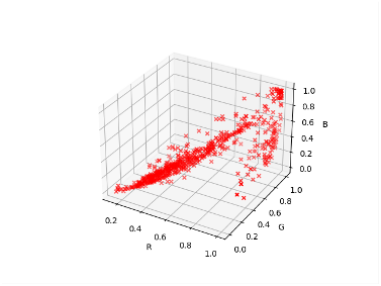
5. Matting

The process of Matting allows the separation between the foreground and background of the object in focus. It involves the process of solving for the foreground opacity $\alpha(x, y)$ and a pixel (x, y). This is given by the matting equation:

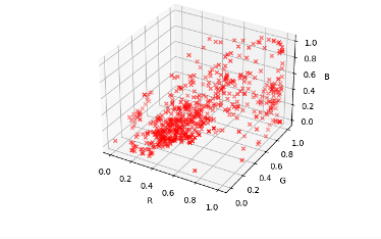
$$I(x, y) = \alpha(x, y)\mathbf{F}(x, y) + (1 - \alpha(x, y))\mathbf{B}(x, y)$$

Where $I(x, y)$ is the image composed of the foreground component $\mathbf{F}(x, y)$ and background component $\mathbf{B}(x, y)$. It can be assumed that $\alpha(x, y)$ is aligned between the RGB planes. Based on the observed image $\mathbf{I}$ Solving this equation becomes an under-constrained problem because we have 3 knowns: I_r, I_g, I_b and 7 unknowns: α, F_r, F_g, F_b, B_r, B_g, B_b at each pixel (x, y). Therefore, to obtain further constraints on this problem a trimap can be used, coming from the computed depth map in the previous steps. A trimap, shown in [Figure 4] has 3 regions: a known region which is strictly foreground, a region which is strictly background and an unknown region which consists of pixels that are in between the foreground and background regions.The difference in misalignment between the foreground and background can be then used to optimize the matte to obtain a region with a very small unknown to tightly segment the image between foreground

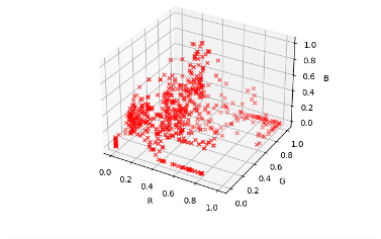
and background.



$d = 0$, $L = 0.02$



$d = 3$, $L = 0.66$



$d = 5$, $L = 0.75$

Figure 3: Cluster of pixels within the window outlined with the red square.

Algorithm 1: Matte Optimization Algorithm

Initialization

1. Construct a trimap from the disparity.
2. Find an initial matte α based on the trimap.
3. Obtain foreground and background disparity maps d_F and d_B

while *Not Converged* **do**

1. Estimate foreground and background color F_n, B_n based on the current α
2. Compute consistency measures C_{F_n} and C_{B_n}
3. Update α_{n+1} based on C_{F_n} and C_{B_n}

end

5.1. Matte Optimization Flow

To optimize matte in an iterative method to obtain the optimal segmentation between the foreground and background, various consistency methods are used based on the Matte Optimization Algorithm [2] shown in Algorithm 1. The initial step for initialization divides the image into foreground and background by thresholding the depth map to create a trimap. The initial α can then be computed from the current trimap. The initial α is computed using a method proposed by [4]. This initial alpha does not do a great job at separating the image between foreground and background in the unknown regions as shown in [Figure 4]. There are many artifacts from the background in the initial matte that should not be included that comes from where the foreground and background colors are similar.

In the iterative steps, first the foreground and background colors are estimated using the current α by minimizing a quadratic cost function based on the matting equation with an added smooth constraint:

$$\sum_{(x,y)} ||\mathbf{I}(x,y) - \alpha_n(x,y)\mathbf{F}_n(x,y) - (1-\alpha_n(x,y))\mathbf{B}_n(x,y)||^2$$

The estimated $\mathbf{F}_n$ and $\mathbf{B}_n$ have the errors in the same location as α_n and the regions of these errors can be detected by measuring how consistent the estimated colors are with the foreground and the background disparity maps $d_F(x,y)$ and $d_B(x,y)$. Then the corrected α_n in those regions are updated to update the current α_n to α_{n+1} until convergence when the difference between α_n and α_{n+1} is very small.

5.2. Consistency Measures For Matte

The same approach as using the colors lines model as a correspondence measure can be used in this approach of finding an optimal matte. Utilizing a small window of pixels in the foreground image $\mathbf{F}(x, y)$ as a set of pixels as $S_F(x, y; d)$ with an estimated disparity d, it can be expressed as: $S_F = \{(F_r(s + d, t), F_g(s, t - d), F_b(s - d, t))|(s, t) \in w(x, y)\}$ Therefore, the foreground color lines model error can be expressed as:

$$e_F(x, y; d) = \frac{1}{N} \sum_{i=1}^{N} l_i^2$$

$$e_B(x, y; d) = \frac{1}{N} \sum_{i=1}^{N} l_i^2$$

Where $N = |S_F(x, y; d)|$, l_i is the distance of the i-th color within the window of pixels $S_F(x, y; d)$ from a line fitted to the cluster of pixels. This approach is to examine if colors within the window $S_F(x, y; d)$ fit the colors lines model. As a result as $e_F(x, y; d)$ gets larger, the disparity d becomes large as well meaning that it is not the correct disparity. These equations are equally represented for the background image $\mathbf{B}(x, y)$ as well.

The color lines model error can be decomposed further into a simpler equation. For the foreground case, by letting c_i be the i-th color in the set of pixels in the foreground pixel window $S_F(x, y; d)$, with μ being the mean color, v_0 a unit vector of the fitted line, we can get the distance l_i of a point c_i from the line as:

$$l_i^2 = |c_i - \mu| - ((c_i - \mu)^T v_0)^2$$

The average of the first term, $|c_i - \mu|$, is the variance, which can be expressed as:

$$\frac{1}{N} \sum_{i=1}^{N} |c_i - \mu| = \sigma_r^2 + \sigma_g^2 + \sigma_b^2$$

The average of the second term, $((c_i - \mu)^T v_0)^2$, will equal to:

$$\frac{1}{N} \sum_{i=1}^{N} ((c_i - \mu)^T v_0)^2 = v_0^T \left(\frac{1}{N} \sum_{i=1}^{N} (c_i - \mu)(c_i - \mu)^T \right) v_0$$

$$= v_0^T \Sigma v_0 = v_0^T (\lambda_0 v_0) = \lambda_0$$

Therefore, by the definitions of the covariance matrix Σ of the set of pixels in a window of the foreground image $S_F(x, y; d)$ and the eigenvector v_0, the color lines model error can be expressed as:

$$e_F(x, y; d) = \sigma_r^2 + \sigma_g^2 + \sigma_b^2 - \lambda_0$$

The same equations can be applied to the color models error for the background image $e_B(x, y; d)$ as well. However, estimation of the background disparities are are larger than foreground disparities and as a result $e_B(x, y; d)$ is discounted by 0.9.

Now, the color consistency measures can be derived by using the two color models error line model $e_F(x, y; d)$ and $e_B(x, y; d)$ along with the two possible disparities $d_F(x, y)$ and $d_B(x, y)$ at each pixel (x, y) in the unknown region:

$$C_F(x, y) = \exp\{\frac{e_F(x, y; d_F) - e_F(x, y; d_B)}{k_s}\}$$

$$C_B(x, y) = \exp\{\frac{e_B(x, y; d_B) - e_B(x, y; d_F)}{k_s}\}$$

Where k_s is a scale parameter. This consistency equation considers the shifted disparity within a given window to compute the color lines model error. If the estimated foreground color $\mathbf{F}_n$ around (x, y) contains the true background color, then $C_F(x, y)$ will be large around that region because the color lines model error $e_F(x, y; d_F)$ for the foreground image with the foreground disparity d_F will be large while $e_F(x, y; d_B)$ for the background disparity d_B will be small. These same principles can be applied to the background consistency measure $C_B(x, y)$.

5.3. Solving for Matte

This next subsection will discuss how to solve for the next iteration of matte $\alpha_{n+1}(x, y)$. $\alpha(x, y)$ can be solved using as a soft-graph labeling problem similar to [6]. Each pixel in the $\alpha(x, y)$ image can be represented as a node in the graph where each pixel has associated data weights for the foreground and background, along with edge weights. The data weights for the foreground $W_F(x, y)$ pulls α towards 1 while the data weights for the background $W_B(x, y)$ pulls $\alpha(x, y)$ towards 0. The edge weights $W_e(x_0, y_0; x_1, y_1)$ enforce a spatial smoothness to

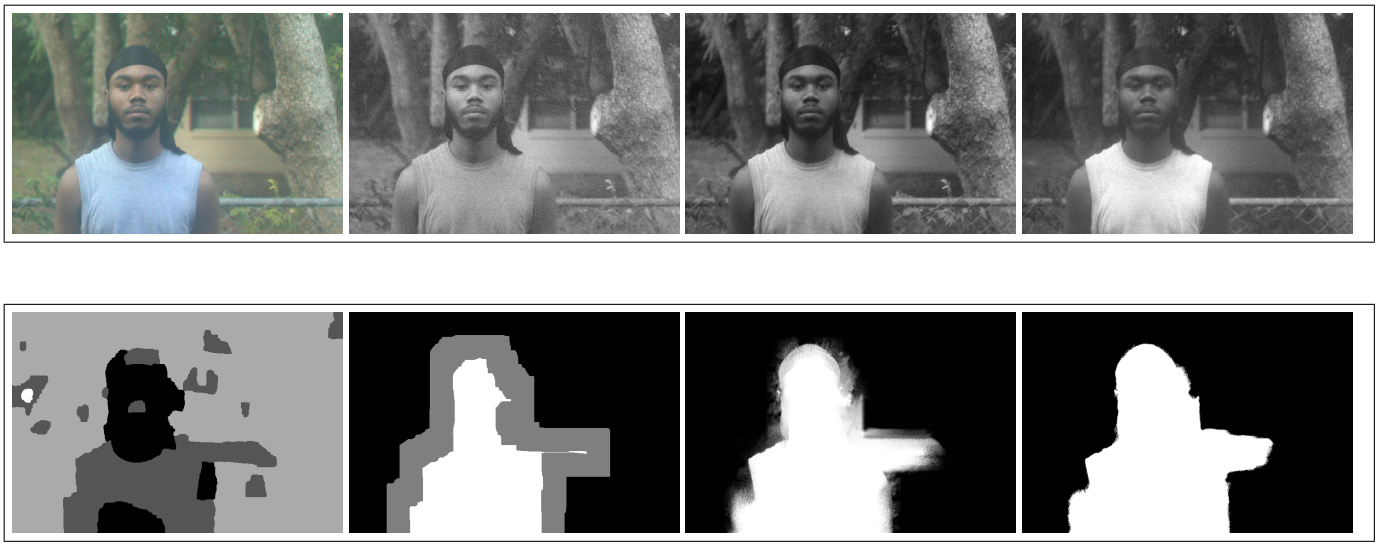


Figure 4: (TOP - From left to right): Captured Image and corresponding Red, Blue, and Green Channels. (BOTTOM - From left to right): Corresponding depth map, trimap, initial matte, optimized matte.

create a tight smooth bound between the foreground and background alpha values. The edge weights are computed using the Matting Laplacian method [4]. This can then all be solved as a sparse linear system while ensuring that $\alpha(x, y)$ remains within the range [0, 1].

The iterative step of Algorithm 1 updates the data weights $W_F(x, y)$ and $W_B(x, y)$ according to the consistency measures $C_{F_n}(x, y)$ and $C_{B_n}(x, y)$ as:

$$W_{F_n} = k_\alpha \alpha_n(x, y) + k_c \left(C_{B_n}(x, y) - C_{F_n}(x, y)\right)$$

$$W_{B_n} = k_\alpha (1 - \alpha_n(x, y)) + k_c \left(C_{F_n}(x, y) - C_{B_n}(x, y)\right)$$

Where k_α and k_c are constants, these background and foreground weights iteratvely improve $\alpha(x, y)$ by removing the background artifacts in the unknown region to better segment the foreground and background α. The foreground weights $W_{F_n}(x, y)$ are clipped at 0 to prevent negative numbers. An analysis of the foreground and background weights shows that when the foreground consistency measure C_{F_n} is smaller than the background consistency measure C_{B_n}, the foreground data weight W_{F_n} increases while the background data weight W_{B_n} decreases. As a result $\alpha_{n+1}(x, y)$ becomes closer to 1 compared to $\alpha_n(x, y)$. Adversely, $\alpha_{n+1}(x, y)$ becomes closer to 0 if C_{B_n} is larger than C_{B_n}.

With this step, we can now update $\alpha(x, y)$ iteratively until convergence until the optimal $\alpha(x, y)$ matte is computed for the image to create a seamless smooth segmentation between the foreground and the background image.

6. Results

Sample images from my implementation are included in the submission folder as well as in this paper as well. The sample images include:

- The image captured with the RGB color-filtered aperture.
- The corresponding depth map
- The trimap
- The original α matted image
- The optimized α matted image
- The foreground disparity d_F image and the background disparity image d_B
- The foreground image placed on a different background
- The image with a change of focus between the foreground and background.

My code was written in the Python programming language. I used images captured by myself along with images captured from the original author of the paper to compare my results.

The parameters that I used in my implementation are as follows:

- $k_s = 0.1$
- $k_\alpha = 0.01$
- $k_c = 0.02$
- Varied the estimated disparity range from -5 to 10
- 20 iterations for converge for the Algorithm 1

The window size for my implementation was 10×10 with a stride of 10. The results of my implementation was similar to the original paper in terms of computing the local depth-map. I could not find an energy minimization framework that utilized graph-cuts in python that could effectively compute an accurate depth-map. Therefore, I utilized the original source code to compute the global depth-map for an image.

I noticed that although my implementation of computing the optimal α matte worked effectively, in some regions the background was included in the α matte. However I was still able to compute many post editing effects on the image.

[Figure 5] and [Figure 9-13] shows a sample image from capture to post edit.

7. Limitations

For the overall research, the limitation of course is capturing an image that is strictly either red, green, or blue when capturing the image using the color-filtered aperture. During this issue, the algorithm cannot compute depth from misalignment because misalignment cannot be computed through one channel. Combining depth from defocus methods could help in this case or even using a neural network with trained data on depth from defocus can help in these situations.

Using color filters in the aperture of the lens also decreases the amount of incident light being received to the camera sensor which requires an increase in the aperture size which increases more defocus.

Matting does not do very well in which the foreground and background colors are similar. I noticed this when attempting to extract the matte from a brown toy bear against a brown wall.

My personal limitations during my implementation was initially the color-filtered aperture. The quality of the gel filter was low and scratched very easily when attempting to put it together. Therefore it created a very noisy image when captured.

I also could not get a perfect fit of the color-filtered aperture inside of the lens of the DSLR camera and as a result I had to tape it and be very difficult. I believe this shifted the blue position of the color filter which contributed the lack of a right-shift in the blue direction. However, I was still able to compute decent depth mapping and alpha matting in certain instances with my implementation.

8. Conclusion

In conclusion, for this project, I used a color-filtered coded aperture to compute depth from color misalignment and further used an optimized α matting optimization algorithm to segment the image between foreground and background. By arranging the square RGB color filters in an arrangement on the lens aperture such that there is a right-shift in the red channel, up-shift in the green channel, and left-shift in the blue direction, depth from image misalignment can be computed. This approach is much better than other approaches for extracting depth because this measure captures depth from a single exposure, it is inexpensive, easy to build, as well as compact. There can be many improvements to this method however. One is to alter the arrangement of the color-filter aperture or even utilizing different colors to represent different wavelengths. In fact, many current studies are experiment with different methods of using various color filters to extract depth from a single exposure.

References

[1] Y. Amari and E. H. Adelson. Single-eye range estimation by using displaced apertures with color filters. In *Proceedings of the 1992 International Conference on Industrial Electronics, Control, Instrumentation, and Automation*, pages 1588–1592 vol.3, 1992. 1

[2] Yosuke Bando, Bing-Yu Chen, and Tomoyuki Nishita. Extracting depth and matte using a color-filtered aperture. *ACM Trans. Graph.*, 27(5), Dec. 2008. 3

[3] Anat Levin, Rob Fergus, Frédo Durand, and William T. Freeman. Image and depth from a conventional camera with a coded aperture. *ACM Trans. Graph.*, 26(3):70–es, July 2007. 1

[4] A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):228–242, 2008. 3, 5

[5] I. Omer and M. Werman. Color lines: image specific color representation. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–II, 2004. 2

[6] Jue Wang and Michael Cohen. Optimized color sampling for robust matting. pages 1–8, 07 2007. 4

9. Appendix

This appendix will include comparisons of my implementation to that of the original paper in terms of trimap, original matting, and matting optimization. This section will also include post editing effects after obtaining the optimal matte for an image.

Figure 5: Dog toy example captured with color-filtered aperture.

Figure 6: Left - Author's Trimap. Right - My Trimap.

Figure 7: Left - Author's initial matte. Right - My initial matte.

Figure 8: Left - Author's optimized matte. Right - My optimized matte.

9.1. Post-Editing Effects

Different Background

Figure 9: Image captured with color-filtered aperture.

Figure 10: Segmented Image using my implementation of the Matting Optimization Algorithm

Figure 11: Focused image placed on beach background.

Artificial Aperture

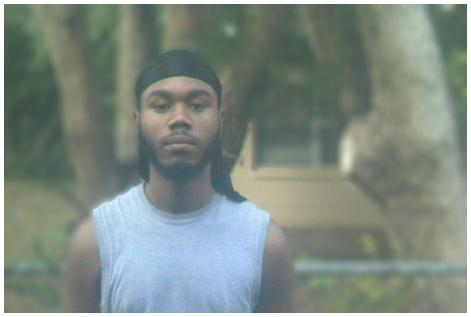


Figure 12: Background blurred with human in the focus plane.

Figure 13: Foreground blurred with background in the focus plane.

9.2. Construction of the Color-Filtered Aperture

Figure 14: Construction of the Color-Filtered Aperture