Multi-Flash Imaging: Techniques for Edge Detection and Non-Photorealistic Rendering

Kavya Tummalapalli
Computational Photography
Carnegie Mellon
ktummala@andrew.cmu.edu

Abstract

In this project, I will loosely be following the approach of Raskar and Yu in their paper ***Non-photorealistic Camera: Depth Edge Detection and Stylized Rendering using Multi-Flash Imaging***[1]*. Through this, I will be creating a hardware prototype of a camera with multiple flashes that are positioned to cast shadows at different angles along the scene.*

Edge detection is a vital part to generating animated and stylized images, where edges are outlined and minute details are suppressed. Using the setup, we can detect depth discontinuities at all angles and use this edge detection to create stylized rendering of images.

The use of multiple flashes provides a new method to edge detection and stylized rendering that does not require a 3D model and provides us with an efficient and cheap technique to produce similar quality images.

1. Introduction

The goal of the non-photorealistic rendering technique is to outline objects and reduce the visual clutter of details. This can serve many important purposes such as during endoscopy or with complex with low contrast, and can also provide an aesthetic look to individual's personal or business photography needs. Many filters on social media exist today that replicate similar techniques of stylizing, but it is intriguing to see how this multi-flash setup compares and contrasts to methods with one or no flashes.

1.1. How it Works

This process is based on observations of epipolar shadow geometry as shown in Figure 1. The light rays originating from P_k are represented as the epipolar rays e_k. We find that a shadow of a depth edge pixel lies on the epipolar ray passing through the pixel and and we only observe the shadow on the background pixel on the side that is opposite of the depth edge along the ray. So as Figure 1(b) shows, the orange area would be shadowed by image 1 taken with the P_1 flash, while the images taken with P_2 and P_3 would not shadow the same area.

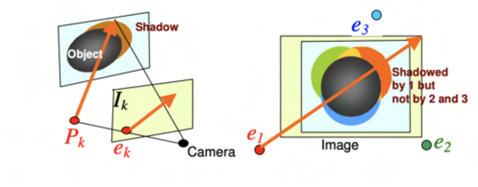

Figure 1. Figure from [1]. The shadows of the object are cast along the epipolar ray. Various epipolar rays cast shadows in different directions and the multi-flash ensures all sides of the object are shadowed in at least one image and lit in at least one image.

1.2. Building the Prototype

To create the prototype, I acquired four flashes (Neewer TT560 Speedlite Flashes) and a DSLR camera (Nikon 3500). Using a square box so the flashes were evenly spaced, and loads of duct tape to secure the flashes, I created a setup of the flashes in the desired four directions as shown in Figure 2(a). As this was not simply mountable to the camera as the weight was very uneven, I created a setup of table and books to hold the flashes in place right behind the desired location of the camera as shown in Figure 2(b). The main challenges in creating this prototype were that the flashes were heavy and I was unable to mount them to the camera itself, so I also had to make sure the flashes were evenly spaced from the lens and untouched during the whole imaging process.

2. Edge Detection

The basic process we will be using for edge detection is by detecting shadows across the four directional flash images. By having four flash images at different sides, we aim to capture shadows along the entirety of the objects in the

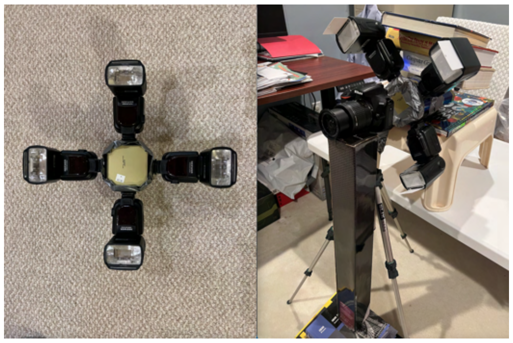
Figure 2. (a) My prototype of a multi-flash setup with flashes evenly spaced in four opposite directions. (b) My multi-flash stylized imaging camera setup that captures images under four different flash conditions.

scene and by extracting these shadows from the rest of the image, we can develop a clean detection of the edges in the image versus other methods that only have access to a single image.

2.1. Overview

Once we have created our camera setup, the general algorithm is as follows:

- Capture an ambient image of the scene.
- Capture four flash images with the top, right, bottom, and left flashes respectively.
- Compute the max image (image with no flashes) from the max pixel values of the four flash images.
- Create four ratio images comparing the flash images to the max image as (flash image / max image).
- For each of the ratio images, use the Sobel filter in the respective shadow direction to compute the silhouettes in each direction.
- Combine these four silhouettes into a final fused image containing all the confident edges in the image.

2.2. Reliability

The multi-flash edge detection technique is highly reliable in many cases relative to other edge detection methods. The main reason being we have all the information. Other methods that only capture an ambient image or a flash in one direction are attempting to detect depth edges from low contrast or information that is not even visible in the specific image, even if it visible to the eye based on our knowledge of the scene. With the multi-flash technique, however, we are supplied multiple shadowings of the object from various directions, allowing us the extract a shadow “silhouette” of the image. By doing so, we can use this fused silhouette to more accurately detect the edges based on their shadows or depth from the background.

Alas, there are still some conditions where this technique may falter. These conditions can lead to either missed depth edges or create extra noise distracting away from the actual edges in the image and thus we must set our environment up to either account for these mishaps or create modifications to the algorithm to fix these discrepancies. Missed edges can be a result of low contrast or discontinuous shadows, while curved surfaces or specularities on the object could lead to extra noise in the depth edge image. To combat this, I made sure to take pictures with a high contrast background and used objects in the scene that weren’t extremely reflective.

3. Image Rendering

The image stylization process consisted of separately creating an image which dulled out the details and features of the object. By doing so, in combination with the edge detection process, we could achieve the cartoon-like look to the images.

I experimented with multiple techniques. First, I tried to implement the procedure from the paper [1]. However, this was a very complicated process that in the end I was not able to fully replicate so to at least achieve a similar look, I turned to other methods.

The next method I tried was the use of a Gaussian blur filter which helped me achieved a pretty decent that was comparable to an abstraction filter. Though not exactly the same, it did help get rid of minute details in the objects, which helped accentuate the edges in the stylized images.

The last method I tried was to implement was using openCV. OpenCV provides many stylization filters, including a filter called edgePreservingFilter. This achieved a look more similar to the paper as the middle was blurred like my simple blur filter, but the edges were still sharp to capture the depth edges well.

I experimented with multiple objects and lightings: displayed here are the bear with ambient ceiling lighting (Figure 3) and elephant with a dark room (Figure 4).

4. Results

My approach started with using the source images from the source paper [1] and making sure my code achieved similar results to those in the paper to check for the correctness of my algorithm (Figure 5). Only after that did I turn to taking my own pictures as I was unsure if they would hold the same quality.

My assumption was somewhat right as the source images achieved much sharper shadows than I could in the multiple lighting environments that I tried. The edge detection did work somewhat smoothly, but due to some displacement discrepancies with my family’s tripod and variability in the timing of the flashes, some shadows/edges were much more easily detected than others (See comparison of intensities in Figure 6). This resulted in uneven edges or those that were a few pixels next to rather than right on the object itself.

If I had more time and resources, my next steps would be to try to improve on my prototype by creating a 3D printed prototype holder for the flashes that could hold them in place on the camera and also look into smaller flashes to make the prototype more sustainable and move less. I would also experiment with more objects and ambient lighting techniques as my house has similar lighting throughout so my only options were to turn on or off the light.

5. Conclusion

Through the work of this project, I have been able to prototype by own multi-flash camera and use it to detect depth discontinuities without relying on 3D reconstructions. In conjuction with the most recent assignments, I found this very fascinating to compare the benefits and drawbacks of each depth-related technique, with the multi-flash camera being very efficient to detect the depths and extract the object, but faltering on 3D aspects as we were using still images rather than a video involving movement.

Using NPR techniques, I was able to render stylized photos and create cartoon-like image. Though my prototype was hefty and temporary, this process could easily be implemented permanently into cameras or smartphones with already existing reasonably sized flashes and it would be intriguing to see if this will ever be considered to be part of a product one day to easily use multi-flash for various purposes.

Acknowledgements I would like to thank the 15-463 Computational Photography staff for their great help and guidance throughout the semester! :)

References

[1] Rogerio Feris Jingyi Yu Matthew Turk Ramesh Raskar, Kar-Han Tan. *Non-photorealistic camera: depth edge detection and stylized rendering using multi-flash imaging*, 2008. In Proceedings of SIGGRAPH 2004, ACM SIGGRAPH. 1, 2

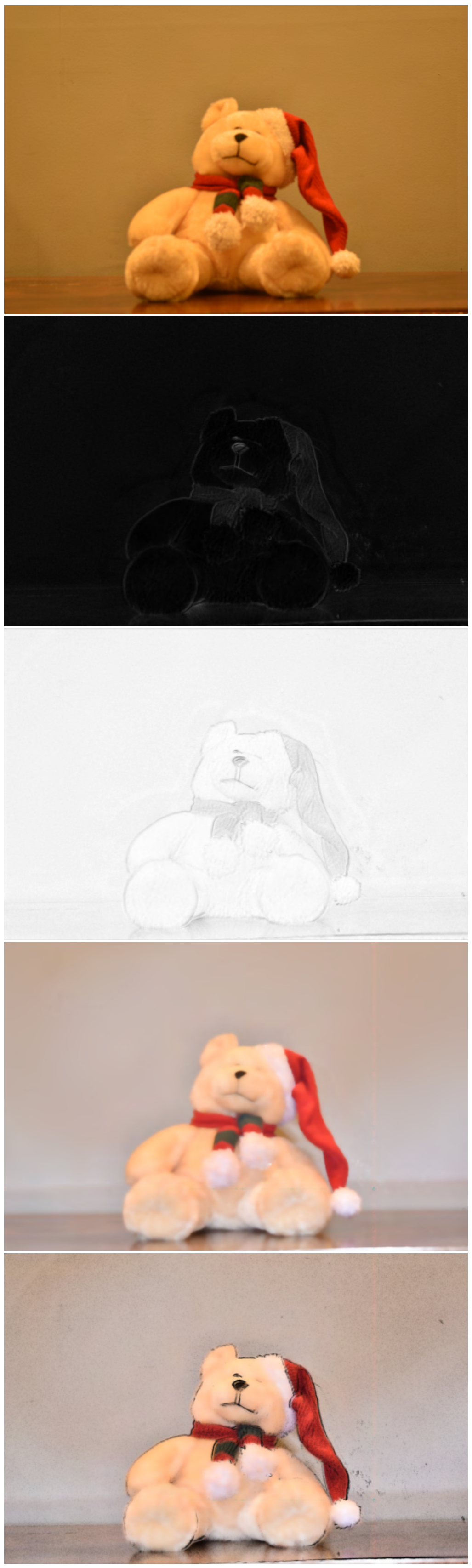
Figure 3. Bear: (a) Ambient image capture (b) Computed depth edges (c) Edge map to be combined with abstracted image (d) Abstracted image using Gaussian blur (e) Fused stylized image from c and d.

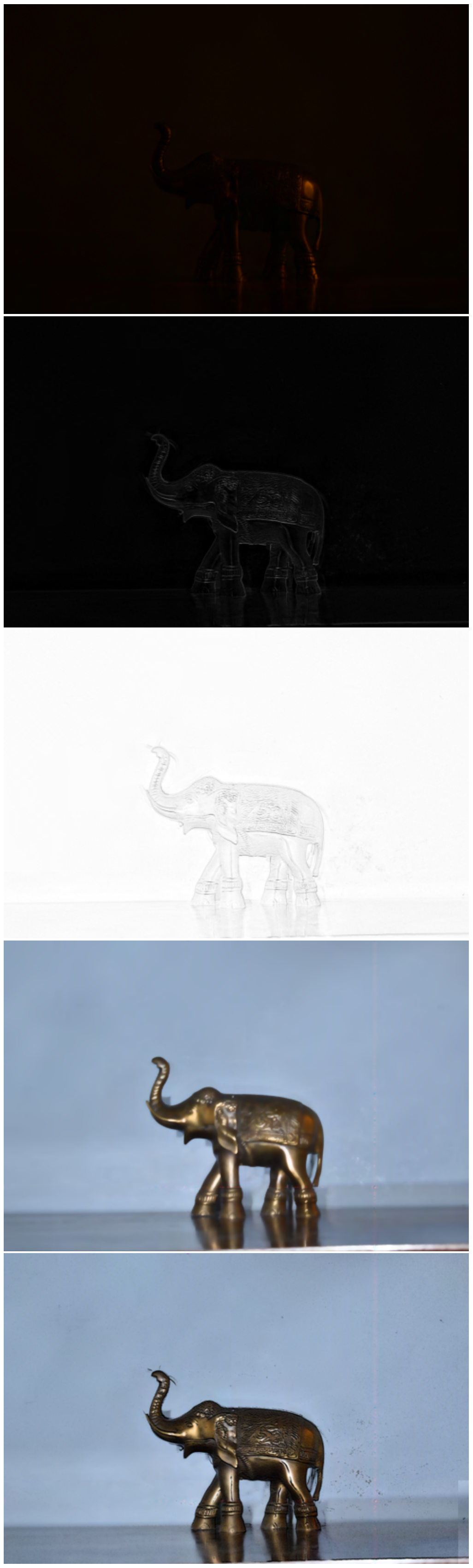
Figure 4. Bear: (a) Ambient image capture (b) Computed depth edges (c) Edge map to be combined with abstracted image (d) Abstracted image using Gaussian blur (e) Fused stylized image from c and d.

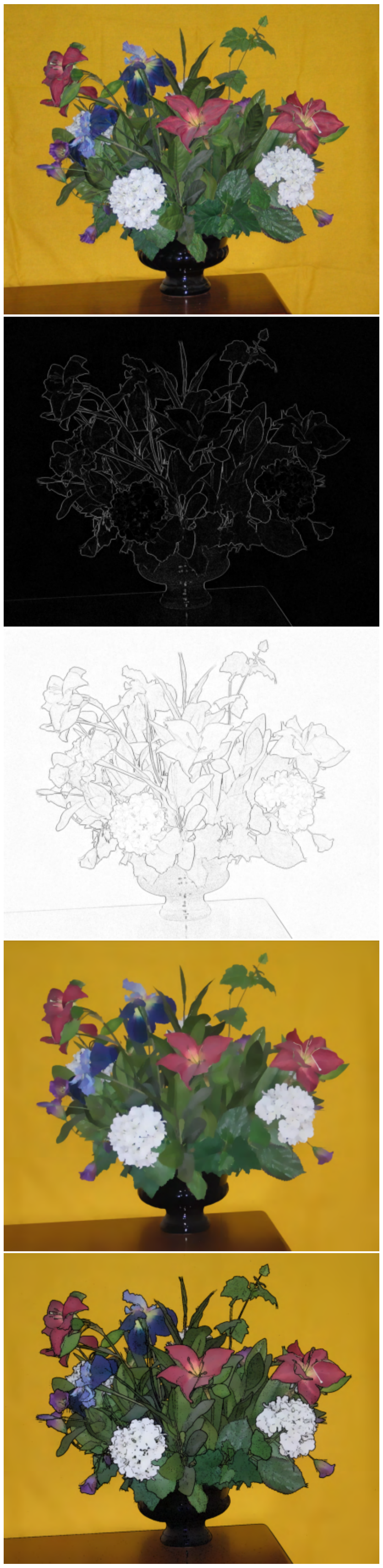
Figure 5. Flower: (a) Ambient image capture (b) Computed depth edges (c) Edge map to be combined with abstracted image (d) Abstracted image using Gaussian blur (e) Fused stylized image from c and d.

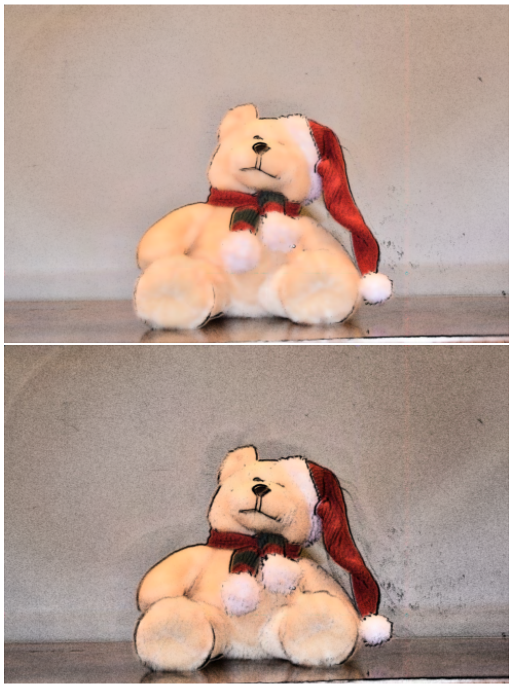

Figure 6. Low and high intensity depth detection. The former produced less noise but missed some edges of the bear, while the latter produces a solid outline of the bear as desired, but also brings in a lot of noise to the background.