Reflection Removal on Mobile Devices

Yilong Geng
Department of Electrical Engineering
Stanford University
Email: gengyl08@stanford.edu

Zizhen Jiang
Department of Electrical Engineering
Stanford University
Email: jiangzz@stanford.edu

***Abstract*—Reflection removal is widely needed with the prevalence of camera equipped mobile phones while no available algorithm is ready for this application. In this paper we propose our reflection removal algorithm which is specifically designed for smart phones. Our algorithm requires the phone app to take two pictures of the same target, one with the flash light on and one with the flash light off. Then we will find the flash spot in the picture, do feature matching to align the pictures, do color transformation on the picture and finally combine the two pictures. Through these steps we will get a resulting image with reflections removed as much as possible.**

I. Introduction

Removing specular reflections from images has been studied by computer vision community in many cases. In computer vision, removing reflections is a very useful pre-processing step to remove undesired information from the image to be analyzed. To reach this goal many algorithms have been developed. These algorithms fall into one of several categories: Lambertian assumption based, color based, polarization based and ICA (Independent Components Analysis) based. For example one ICA based reflection removal algorithm needs pictures taken through a linear polarizer at different angles such that these pictures are linear combinations of the original light and the reflected lights of different weights. The original light information is then acquired by Principle Component Analysis.

Besides computer vision, reflection removal is also widely needed in our daily lives, especially with the prevalence of camera equipped smart phones. With their mobile phones, when people take pictures of clothes behind windows, paintings behind glass covers or figures on computer screens, the reflections always severely degrade the quality and usability of the resulting pictures. What is worse, none of the algorithms above can be used to resolve this issue. The algorithms processing a single picture just don't have enough information to reconstruct the image behind the glass. The mission of taking differently polarized pictures is impossible for mobile phones. It is also impractical to ask the users to take pictures of the target from different angles for our algorithm to process because few users would do that in real circumstances due to its complexity.

In this paper we will introduce our reflection removal algorithm designed for camera equipped mobile phones. Our algorithm requires the user to take two pictures that one is taken with the flash light open and the other is taken with it closed. After the two pictures are taken our algorithm will combine them to produce a new picture with reflections removed as much as possible. One thing to note is that our algorithm don't require the two pictures are taken from the same angle or from different angles. This feature makes the end users' operation as easy as taking only one picture. The users just need to tap once on the trigger button, and the app will take two pictures consecutively, with the flash light on only for the second picture.

Our algorithm consists of a sequence of operations. In the second section we will analyze the characteristics of the two source pictures and explain the overall logic and flow of our algorithm. The third section explains how we identify and characterize the flash spot in the picture taken with flash light on. The fourth section shows how we match the two source pictures before combining them. The fifth section will carry out the method we use to transform the color of one source image to reduce the effect of different lightings. The Sixth section talks about how to combine the two pre-processed source images to make the resulting image seem natural. Lastly, the seventh section will conclude the pros and cons of this algorithm.

II. Algorithm Logistics

Figure 1 shows an example of two input pictures for the algorithm. The two pictures were taken in an indoor environment during the night, from two slightly different angles. Figure 1a is the picture taken with the flash light off. As we can see there are extensive reflections on the glass, which severely degraded the quality of the picture. Figure 1b is the picture taken with the flash light on. Instead of the original reflections, there is a bright flash spot in the picture. Although the rest of this picture is clear and clean, it is also highly defected thus not directly usable.

As we have seen, the quality of Figure 1b is much better than that of Figure 1a and satisfies the requirement of reflection-free except the bright flash spot area. If we can extract the lost information of the white spot area from Figure 1a to compensate Figure 1b, hopefully we can get a resulting picture with reflections removed as much as possible. However, there are several obstacles in the way. First, we need to find out where and how big is the flash spot in flash-light-on picture. Second, we need to align the two input pictures because they can be taken from different angles. Third, the different light sources cause color inconsistency in the two pictures. We need to transform the color of one picture to

(a) Picture taken with flash light off

(b) Picture taken with flash light on

(c) Result image with reflections removed

Fig. 1. Two input pictures and algorithm output

match that of the other picture before combining them. Lastly, if we hard-combine the flash-spot area of Figure 1a and the rest of Figure 1b, the resulting image would seem artificial. We would need to figure out how to soft-combine the two pictures to get a natural looking resulting image.

III. Flash Spot Characterization

Since our strategy is to use Figure 1a to compensate the defected area of Figure 1b, the first step is to find the position and size of this defected area–the bright flash spot.

A. Flash Spot Positioning with Matched Filter

By observing many images with flash spots, we can find that the patterns of the flash spots are very similar–very bright in the center and fade away as deviating from the center. This similarity suggests that matched filter is a good method to find out the position of the flash spot in the picture taken with flash light on.

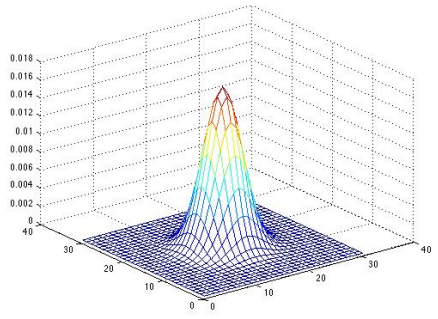

Fig. 2. Gaussian Matched Filter

What matched filter do we use? Still by observing the flash spots, we think a Gaussian filter like in Figure 2 could be a good approximation of their intensity pattern. The next question would be how do we decide the size (or σ) of this Gaussian filter. Since it is impossible to know the size of the flash spot before hand, it is not likely that we can find a Guassian filter that matches the flash spot's size. However, by experiments we found that if we constrain the σ of the Gaussian filter to be between 0.005 to 0.05 of the width of the picture, it does not really affect the matching result in real applications. So in our algorithm we just set $\sigma = 0.01 * Width$.

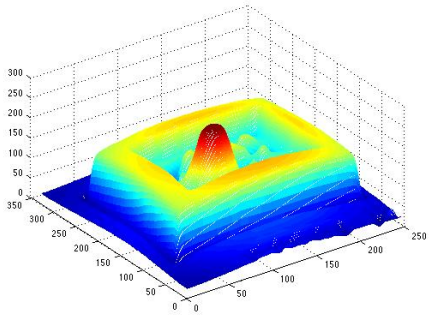

Fig. 3. Matched Filter Result

Figure 3 shows the filtering result of the Gaussian matched filter. By picking the position of the highest value in this respones, we can find the center of the flash spot.

B. Flash Spot Size Characterization

After knowing the position of the center of the flash spot, it is time to get the diameter of it. Since the flash spot fades away from its center gradually, the diameter we talk about here is its equivalent diameter.

Since the flash spot is always very bright in the image, first we binarize the image with a threshold of 0.95 to remove as

many other bright areas as possible. In the second step we get the connected components of the binarized image, and pick out the connected component that contains the flash spot center we have already found. Lastly, by getting the equivalent diameter d of this selected connected component, we can make some sense of the size of the flash spot. The equivalent diameter we get here will be used in the color transformation section and pictures combination section.

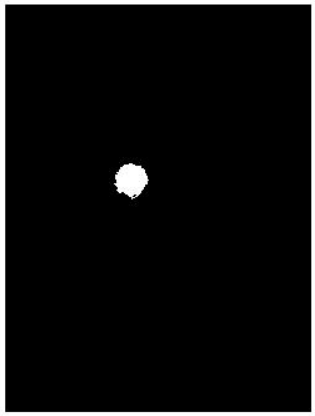

Fig. 4. Connected Component Containing Flash Spot Center

IV. Feature Extraction and Matching

Since the two input pictures can be taken from different angles, we would need to align them before trying to combine them. The algorithms used here are pretty standard: SIFT descriptors and RANSAC with homography model.

A. Model Learning

Fig. 5. SIFT Descriptors and RANSAC with Homography Model

To learn the homography model between the two pictures, we first get the SIFT descriptors of the two input pictures and then learn the model with RANSAC. Figure 5 shows the matched descriptors.

B. Picture Alignment

After getting the homography model, we can project Figure 1a according to the model to make the result align with Figure 1b. Figure 6 shows the resulting aligned image.

Fig. 6. Aligned Reflection Image

V. Color Transformation

After the last two steps, flash spot characterization and image alignment, we are able to actually combine Figure 6 and Figure 1b. However, due to the different light sources of these two pictures, there will be color inconsistency at the flash spot area of the resulting image. So before combining the two picutures we need to transform the color in Figure 6 to make it look consistent to Figure 1b.

Given one point (original point) in Figure 6, how do we transform its color? Since Figure 6 and Figure 1b are point to point aligned, one idea could be picking the color of the corresponding point in Figure 1b as its new color. This way the resulting image of color transformation will be identical to Figure 1b. Although this is meaningless, it actually works, except for the flash spot area. This is because except the flash spot area, the rest of Figure 1b is actually desired.

To make color transformation work at the flash spot area, we need to change our algorithm to map a single point in Figure 6 to multiple points in Figure 1b. Instead of picking the color of the corresponding point in Figure 1b, we first find 'neighbor' points in Figure 6 whose colors are close to the color of the original point. Then the new color of the original point is a linear combination of the colors of the corresponding neighbor points in Figure 1b.

Besides the distance in the color space, we also need to consider two other factors in finding neighbor points. The first factor is the neighbor point's distance to the original point on the picture. We would like to emphasize the points that are close to the original point to reduce the effect of uneven lighting in the picture. The second factor is the neighbor point's distance to the flash spot center. Since the flash spot is the undesired information in Figure 1b, we would like the neighbor points to be far away from the flash spot center.

To formalize the algorithm we described above, we define "Neighborness" (N) who indicates the likelihood of

a point being a neighbor of the original point, Where $\sum_{p \in Picture} N(p, p_0) = 1$. Let $Color1(p)$ be the color of point p in Figure 6, $Color2(p)$ be the color of point p in Figure 1b. Then, the new color of the original point can be calculated as

$$Transform(Color1(p_0)) = \sum_{p \in Picture} N(p, p_0) \times Color2(p)$$

The neighborness between point p and point p_0 is a multiplication of three factors:

$$N(p, p_0) = N_{color}(p, p_0) \times N_{dis}(p, p_0) \times N_{flash}(p)$$

N_{color} is the neighborness that describes the similarity of two points in the color space:

$$N_{color}(p, p_0) = Exp(-\alpha_{color} \times Distance(Color1(p), Color1(p_0)))$$

N_{dis} is the neighborness that describes the similarity of two points in terms of their positions on the picture.

$$N_{dis}(p, p_0) = Exp(-\alpha_{dis} \times Distance(p, p_0))$$

N_{flash} is the neighborness used to suppress the white color in the flash spot. It is only a function of the neighbor point and the position of the flash spot center p_{flash}

$$N_{flash}(p) = Exp(-\alpha_{flash} \times Distance(p, p_{flash}))$$

One concern here is that we need to do the computation for every point in the picture. So this step could be computationally expensive. However in reality we only need to do color transformation at the flash spot area. This could help reduce the computation complexity to a acceptable level.

Fig. 7. Image after color transformation

By doing color transformation for every point in Figure 6, we can get Figure 7. We can see that Figure 7 and Figure 1b is very similar in color. Now we are ready to combine these two pictures!

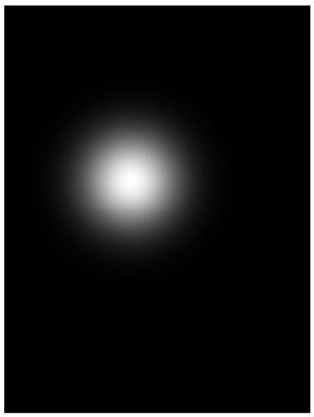

Fig. 8. Mask

VI. Pictures Combination

The last question is how do we combine the two pictures together. As we have discussed, the areas other than the flash spot is desired in Figure 1b. Our plan is to use the flash spot area of Figure 7 to compensate Figure 1b. So with a Gaussian mask like in Figure 8, we can get the combined image according to the following formula:

$$NewImage = Mask \times Figure7 + (1 - Mask) \times FlashImage$$

Fig. 9. Combination Result

Figure 9 shows the resulting combined image. We can see that the mask successively selected out the flash spot area in Figure 7 and the other areas in Figure 1b and added them up smoothly.

VII. Conclusion

Figure 10 is another example of the reflection removal performance of our algorithm. We can see that the algorithm works perfectly as expected.

(a) Picture taken with flash light off

(b) Picture taken with flash light on

(c) Result Image

Fig. 10. A second example

In this paper we have discussed the pipeline of our algorithm. In step 1 we finds out the position and size of the flash spot in the picture with template matching and the connected component method. In step 2 we uses feature extraction and matching to do image alignment. Step 3 resolves the color inconsistancy in the two pictures caused by different lighting sources. And finally step 4 combines the two pictures together to get a resulting image with reflections removed as much as possible.

REFERENCES

[1] H.Farid and E.H.Adelson Separating Reflections from Images Using Independent Component Analysis.

[2] G.Brelstaff and A.Blake. Detecting specular reections using lambertian constraints.

[3] S.Shafer. Using color to separate reflection componetns.