Flash/No-Flash Image Synthesis: Photo Enhancement through Denoising, Detail Transfer, & Color Preservation

Abigail Skerker, Jìa Wilkins

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

May 03, 2024

Technical Report No. ECE-2024-2

Contents

1 Introduction

In low-light settings, capturing images that successfully convey the visual richness of real environments poses a significant challenge in photography. Photographers traditionally have had to choose between utilizing long exposure times, large apertures, increased gain or adding artificial light sources through flash. However, while each of these techniques presents both benefits and drawbacks to capturing fidelitous images, we hope to implement an algorithm to highlight the best and mitigate the worst features of each approach. A list of these attributes with their pros and cons are listed below:

• Exposure Time: by providing more light to the sensor via a longer exposure, more low-light details from the original scene can be captured. However, camera shakes or scene motion during long exposures will result in motion blur, greatly affecting image fidelity.

• Aperture: by letting additional light into the sensor via a larger aperture, a greater level of detail can also be achieved. However, larger apertures are constrained by the physical limits of the lens and also reduce the depth of field, which may not be desired.

• Gain: by applying an amplification factor to all pixels, one can create a brighter image. However, when exposure time is short, the camera cannot capture enough light to accurately estimate the color at each pixel, thus increasing image noise.

• Flash: finally, by adding artificial light sources to the scene, photographers are able to use shorter exposure times, smaller apertures, and lower gains while still capturing enough information to produce sharp, noise-free images. However, objects near the camera are disproportionately brightened, the mood evoked by ambient illumination may be destroyed, and unwanted artifacts such as specular highlights may be introduced from the flash.

2 Literature Review

Given the above constraints, photographers often find themselves limited in their ability to effectively capture every aspect of a scene as desired when working with only a single image. However, if the photographed scene is static or nearly static, two images can be captured one after the other in quick succession – one with flash to capture detail and one without flash to capture ambient illumination. When properly combined, image fidelity can benefit from both the low-noise and high-detail of a flash-exposed picture as well as the natural colors and reduced shadows from the no-flash exposure. This method is outlined by Petschnigg et al. in the 2004 paper "Digital Photography with Flash and No-Flash Image Pairs," wherein they were able to meld flash and no-flash image pairs to produce a final image that accurately communicates the true lighting, details, and overall mood of the original low-light scene [2].

2.1 Bilateral Filtering Variations

Almost every technique for synthesizing higher quality images from flash/no-flash pairs relies on bilateral filtering, a process described in Tomasi and Manduchi's 1998 paper "Bilateral Filtering for Grey and Color Images." A bilateral filter achieves simultaneous noise reduction and edge preservation by applying an edge stopping function to a normal low-pass filter such that the filter weights are lower when the difference in intensity between two pixels is higher. When applied uniformly across color channels, the bilateral filter is able to effect noise reduction and edge preservation for both greyscale and color images [1].

Petschnigg et al defined the filter at each weight as a product of two Gaussian distributions, one dependent on spatial distance (the low-pass filter) and the other on pixel intensity difference (the edge stopping function) [2]. However, in a 2015 paper "Rayleigh based Bilateral Filtering for Flash and No-Flash Images Pairs," the authors describe an alternative approach centered on using a Rayleigh distribution instead of a Gaussian distribution for a bilateral filter, resulting in smoother denoising with the modified filter. Although the Rayleigh distribution itself is not symmetric like a Gaussian, it is here used for the noise reduction aspect of the bilateral filter which is based on the absolute distance

to a pixel. Therefore, using a Rayleigh distribution does not introduce nonlinear phase artifacts, and it instead simply modifies the shape of the decay curve of the filter [3].

Petschnigg et al. also implements a "joint bilateral filter" which applies the same technique as normal bilateral filtering except that its edge stopping function is defined by the intensity difference between pixels in the flash image, not the ambient image. When performing detail transfer, the final image is determined by the product of the noise-reduced ambient image, achieved through the joint bilateral filter, and a detail image produced from the flash image [2].

2.2 Flash Artifact Mitigation

The joint bilateral filter may leave behind artifacts or details produced by the flash image due to the extra light source, namely specular highlights and flash shadows. To mitigate such artifacts, a mask is produced to defer to the bilateral filter of the ambient image in affected areas. After linearizing the image pair and normalizing for ISO and exposure settings, areas of flash shadow should appear identical between the two captures. Therefore, shadow regions can be identified by comparing the two images. If the comparison value is within a small threshold range, a shadow mask is generated. For specular highlights, areas of sensor saturation are detected with a simple luminance threshold.

An alternate approach to handling detail capture and flash shadow mitigation is defined in Eisemann and Durand's 2004 research "Flash Photography Enhancement via Intrinsic Relighting," wherein both the flash and no-flash images are decomposed into color and intensity (large scale & fine detail) images before recombination. Regarding flash artifacts, they first compile a histogram of relative changes in intensity between the flash and no-flash images then compute a threshold for the darkest parts of the shadows. To account for shadow boundaries, they apply the gradients for the intensity in each image such that pixels with a stronger gradient in the flash – that are also close to pixels already determined to be in the umbra – are also considered to be part of the shadow, specifically the penumbra [4].

2.3 Red-Eye Removal

Petschnigg et al. also describes a methodology for handling red-eye correction. Areas of interest are first identified by converting both flash and ambient images to YCbCr chromaticity space, thus separating out red chrominances above a certain threshold in the flash compared to ambient image. By also discriminating for low luminances in the ambient image, red-eye candidates can be determined by searching for elliptically shaped anomalies. Using the same artifact mask for flash highlight and shadow removal, specular highlights in the eye are preserved while the red pixels are replaced with an appropriately luminant gray pupil color [2].

Most of the variety in red-eye mitigation algorithms lie in determining the pupil mask; contemporaneous to Petschnigg et al., Hardeberg's 2001 research describes a technique for correction on only one image. As there is no difference to calculate, an approximate, manually determined region of interest is required first, on which a mask is formed based on "typical red eye color." After being put through several thresholds, refinements, and blurs, a final mask can be obtained for color correction [5]. However, as we discovered in our attempts to capture custom images for this project, most modern-day cameras have non-software red-eye mitigation techniques built in. Specifically, the camera shines a pre-flash for long enough to constrict the pupils before engaging a full flash and capture, thus mitigating red-eye artifacts during the physical flash protocol and eliminating much of the need for post-processing.

3 Problem Statement

Our goal of this project was to implement and test two algorithms included as part of the Petschnigg et al. paper: the Denoising & Detail Transfer Algorithm and the Red-eye Removal Algorithm. For the Denoising & Detail Transfer Algorithm, we aimed to implement a bilateral and joint bilateral filter in MATLAB without using MATLAB's built in bilateral filter as well as implement the detail transfer and shadow and specularity removal algorithm, combining each component in final merged images. Additionally, we identified the impact of different parameters on the algorithm, testing it with both provided images and new images. Lastly, we compared the results we obtained with the results from

the paper. Regarding the Red-eye Removal Algorithm, our aim was to implement a similar process as outlined in the paper to analyze the difference between a flash/no-flash image pair for red-eye artifacts and correct them. The Petschnigg paper went into only limited detail for how they realized their red-eye analysis, so MATLAB's built in functions were utilized to achieve similar results while following the same methodology described for the correction itself.

4 Denoising & Detail Transfer Algorithm Analysis

The Petschnigg et al. algorithm can be effectively summarized in a single block diagram, as shown below [2]:

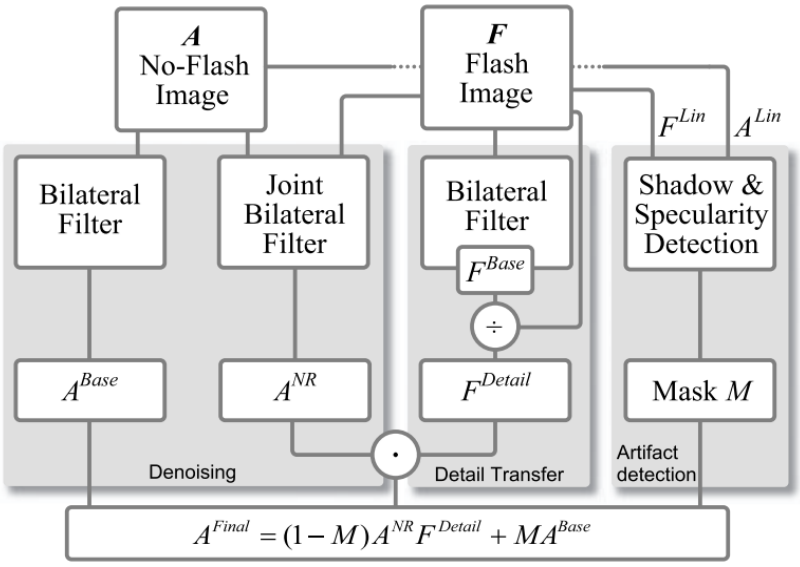


Fig. 1: Denoising & Detail Transfer Algorithm Block Diagram

4.1 Bilateral Filter & Joint-Bilateral Filter

To implement the above denoising and detail transfer algorithm, two filter functions are needed: a bilateral filter and a joint bilateral filter. A bilateral filter is a combination of two filters, a low pass filter and a range filter. A low-pass filter removes noise from an image by performing a weighted average over a neighborhood, where weights are determined by how far two points are from each other. The equation for a Gaussian low pass filter is below in equation 1.

$$A_p^{LP} = \frac{1}{k_d(p)} \sum_{p' \in \Omega} g_d(p' - p)\, A_{p'} \quad (eq. 1)$$

A range filter is an edge preserving filter. A range filter has an almost identical equation to a Gaussian low pass filter, but it determines weights in averaging based on how similar two points in an image are to each other (difference in color value) rather than based on how far they are from each other. The equation for a range filter is below in equation 2.

$$A_p^{Range} = \frac{1}{k_r(p)} \sum_{p' \in \Omega} g_r(A_p - A_{p'})\, A_{p'} \quad (eq. 2)$$

In each equation, k is a normalization constant defined as the sum of the weights for all p. A_p is a point in the image. Ω is the filter, or the area to average over. g(x-y) is the Gaussian function, defined by equation 3 below, and parameterized by the bandwidth parameter σ.

$$g(x - y) = \exp\left(\frac{-\frac{1}{2}(x - y)^2}{\sigma^2}\right) \quad (eq. 3)$$

The bilateral filter is a combination of the domain and range filter above and is shown below in equation 4.

$$A_p^{Base} = \frac{1}{k(p)} \sum_{p' \in \Omega} g_d(p' - p)\, g_r\big(A_p - A_{p'}\big) A_{p'} \quad (eq. 4)$$

The normalization constant is the product of the sum of the weights, defined as:

$$k(p) = \sum_{p' \in \Omega} g_d(p' - p) g_r(A_p - A_{p'}) \quad (eq. 5)$$

The joint bilateral filter is defined almost identically to the bilateral filter except that the range Gaussian is a function of the flash image F rather than the no-flash image A. The reason this is done is because it is assumed that the flash image contains more details than the no-flash image, and so the edges will be better defined.

$$A_p^{NR} = \frac{1}{k(p)} \sum_{p' \in \Omega} g_d(p' - p)\, g_r\big(F_p - F_{p'}\big) A_{p'} \quad (eq. 6)$$

To implement the bilateral filter, we wrote a custom MATLAB function *bilat_filt* with inputs of an image, a magnitude cutoff, a LPF bandwidth (σ_d) and a range filter bandwidth (σ_r). The magnitude cutoff defines the size of Ω, or the size of the filter. The

larger Ω is, the more computationally intensive the algorithm is as more calculations must be performed. However, if Ω is too small, image distortion may occur, as the Gaussian will be cut off. Visually, this is demonstrated in the figure below with σ = 10, where if Ω is of size 41x41, only low-weighted pixels below about 0.02 will be ignored, whereas if Ω is of size 21x21, weighted pixels up to 0.4 will be ignored, which may cause image distortion. The magnitude cutoff in our function defined these corner Gaussian values (i.e. 0.02 and 0.4 in the figure below).

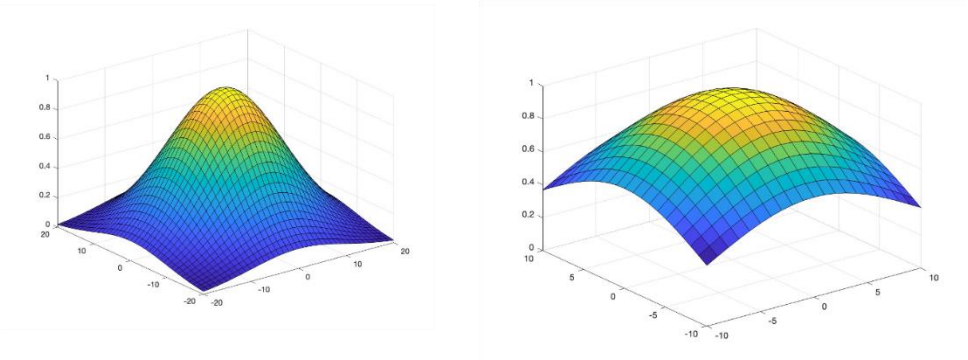


Fig. 2: Gaussian Function of σ = 10 with sizes of 41x41 and 21x21

To implement the joint bilateral filter (*joint_bilat_filt*), we modified the *bilat_filt* function to take in two separate images, flash_img and no_flash image, then modified the code such that the range function used the flash image to calculate the differences in intensity value instead of the no flash image.

4.2 Detail Transfer, Artifact Mitigation, & Final Synthesis

The detail transfer function is defined on the flash image F to produce F^{Detail}. The flash image F is divided by the bilaterally filtered flash image F^{Base} to produce F^{Detail}. To account for low signal noise and divide by zero errors, ε is added to both images and is typically a small constant around 0.02.

$$F^{Detail} = \frac{F + \varepsilon}{F^{Base} + \varepsilon} \quad (eq.7)$$

The flash artifact mask is computed by first linearizing F and A to produce F'^{Lin} and A'^{Lin}, which are then also normalized for ISO values and exposure time (Δt). Image linearization is needed because sensors typically capture light in a non-linear fashion, so

the recorded values do not linearly correlate with the intensity of light hitting the sensor. Linearization is typically performed via gamma correction, where the intensity of light is raised to the gamma value to correct for non-linearities.

$$A^{Lin} = A^{Lin\prime} \frac{ISO_F \Delta t_F}{ISO_A \Delta t_A} \quad (eq. 8)$$

These linearized images are then used to determine a mask of flash shadow locations at identical areas between F^{Lin} and A^{Lin} within a parameter τ_{Shad}:

$$M^{Shad} = \begin{cases} 1 \; when \; F^{Lin} - A^{Lin} \leq \tau_{Shad} \\ 0 \; else \end{cases} \quad (eq. 9)$$

An additional mask of specular highlights from the flash image M^{Spec} defined simply as areas in F^{Lin} with luminosity greater than 95% of the sensor maximum.

$$M^{Spec} = \begin{cases} 1 \; when \; F \geq 0.95 \times L_{Max} \\ 0 \; else \end{cases} \quad (eq. 10)$$

A union of the two produces a final mask M, with a simple blur to eliminate harsh edges.

$$M = M^{Shad} \cup M^{Spec} \; (eq. 11)$$

And ultimately, the final image is produced through merging the above components in the following synthesis equation:

$$A^{Final} = (1 - M) A^{NR} F^{Detail} + M A^{Base} \quad (eq. 12)$$

In our code, we implemented a function *run_algorithm* that took in a flash image, a no flash image, the filter parameters (magnitude cutoff, a LPF bandwidth (σ_d) and a range filter bandwidth (σ_r) for the bilateral filter and joint bilateral filter separately) and a linear scaling factor defined as the ratio of ISO and exposure times. In this function, the bilateral and joint bilateral filters were run on each of the RGB color channel images separately. The detail transfer and mask were then performed, and the final image was synthesized using equation 12.

5 Denoising & Detail Transfer Algorithm Results

As described above, we implemented the denoising and detail transfer algorithm described in the Petschnigg et al. paper using MATLAB. Shown below in figure 2 is a demonstration of this algorithm applied to the "Carpet" image sourced from the paper's dataset [2]. The figure titles match the block diagram in the Algorithm section. Note that to help the algorithm run in efficient time, the images were downsampled from 2048x3072 to 512x768, or by a factor of 4 in each direction.

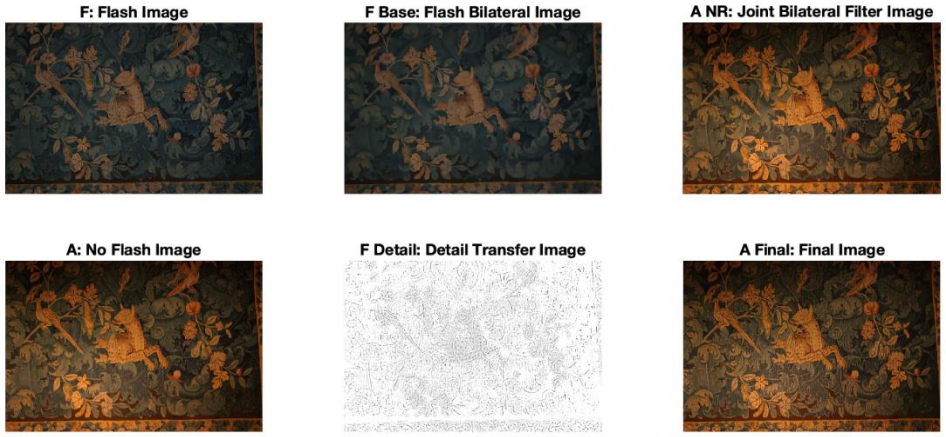


Fig. 3: Demonstration of Detail Transfer & Denoising Algorithm on "Carpet"

Figure 3 offers a zoomed in view to highlight the detail transfer applied in the face of the leopard on the carpet. It can be seen that there is additional resolution on the final image that is not in the original image, while still retaining most of the color from the no flash image as intended by the algorithm.

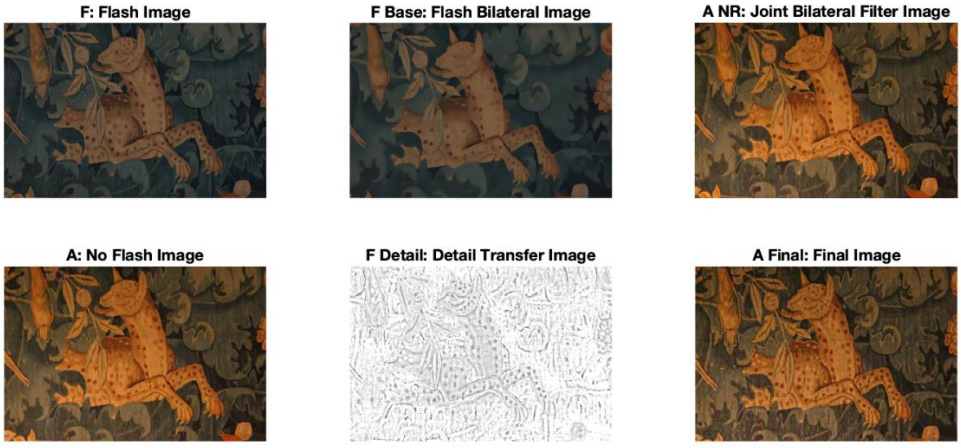


Fig. 4: Demonstration of Detail Transfer & Denoising Algorithm on "Carpet," Leopard

5.1 Impact of Gaussian Weights

Parameters described in the implementation section were modified to assess their impact on image denoising. The bilateral filter is parametrized by σ_d, σ_r. In the below figure, a zoomed in version of "Carpet" is shown bilaterally filtered with three different values of σ_d and three different values of σ_r. While subtle, it can be observed that by decreasing σ in both cases, less blurring occurs. Therefore, the lower bound values of σ were used for denoising in the final result.

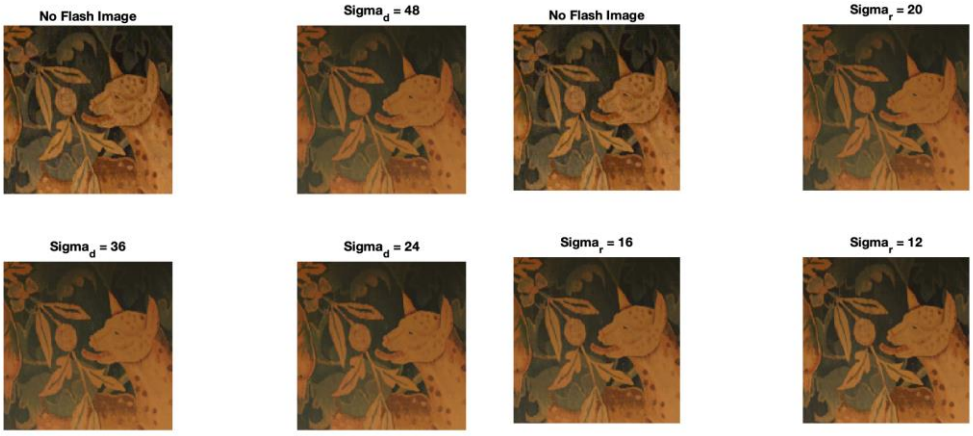


Fig. 5: Effects of Gaussian Weights on Leopard Face, Bilateral Filter

The impact of the Gaussian weights can also be observed on the detail transfer portion of the algorithm, performed on the flash image. While σ_d does not strongly impact the detail transfer portion, σ_r has a strong effect where more details are transferred at a larger value.

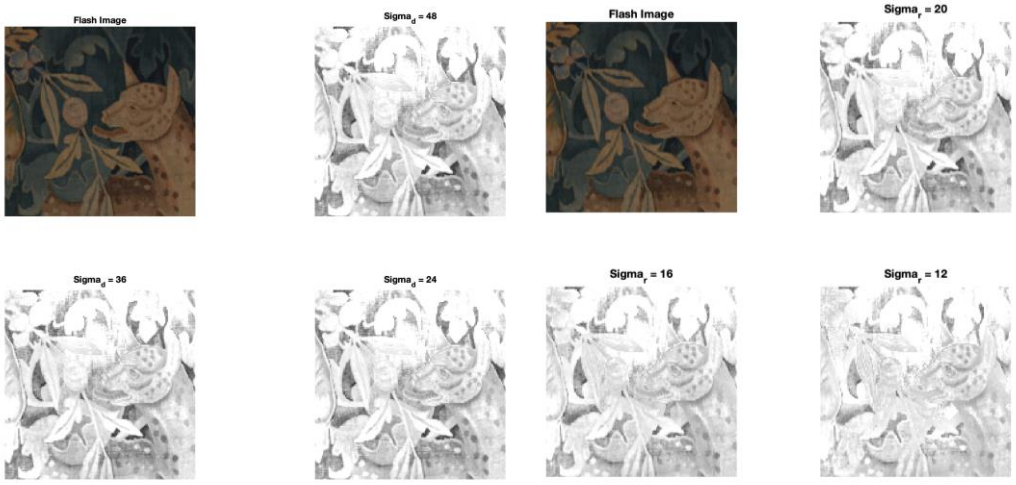


Fig. 6: Effects of Gaussian Weights on Leopard Face, Detail Transfer

In reviewing the final image with varied parameters, it is more challenging to observe the differences between parameters. However, small color variations are noticable.

Fig. 7: Effects of Gaussian Weights on Leopard Face, Final Image Synthesis

5.2 Impact of Filter Size

As discussed in algorithm implementation, it is expected that the size of the filter would have an impact on distorting the final image. In the below figure, the effect of the filter size, as defined by the corner amplitude attenuation of the Gaussian function is shown, where 0.1 would be a larger filter than 0.5. It appears as if small color distortions are present in 0.5 that are not present in 0.1.

Fig. 8: Effects of Filter Size on Leopard Face Color Distortions

5.3 Shadow & Specularity Mask

We implemented the shadow and specularity mask as described in the Analysis section. While we could not identify the exposure time settings from the images in the files, the ISO and exposure times were approximated, and the images were linearized and scaled as such to create the mask.

Fig. 9: Shadow & Specularity Mask on "Carpet"

Although our algorithm was able to successfully identify shadows in the "Carpet" image, the matte surface provides no specular reflections to fully test our algorithm. As will be described later, we used a custom flash/no-flash pair, and our implementation was able to successfully segregate areas of flash shadows specularities.

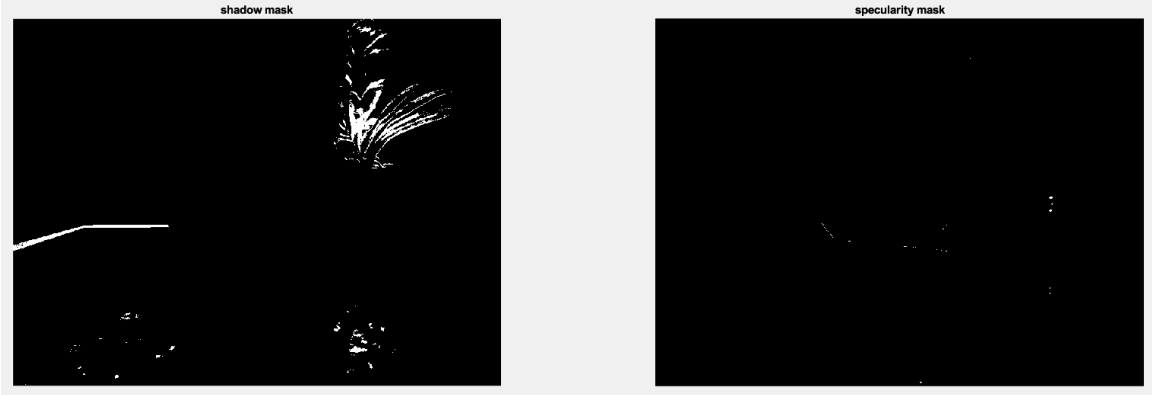


Fig. 10: Shadow & Specularity Mask on Custom Image

Additionally, as we had direct access to our photos' metadata, the ISO and exposure times were known precisely, and as such, linearization could proceed without any approximations. However, we had significantly less control over such parameters, thus making it much more difficult to a achieve the broader spectrum of information seen in the original paper.

5.4 Comparison with Paper Outputs

The paper's dataset provided output images as implemented algorithm, including both the bilaterally filtered no-flash image, and their final synthesized image. We compared the bilaterally filtered images we created to those created by the paper below by creating error images and calculating the MSE. Interestingly and unexpectedly, to achieve the smallest MSE between our images and the paper images (approximately 1.7), we needed to use a very narrow filter width (magnitude cutoff value 0.9). Other parameters such as the σ values affected the difference much less and were run with smaller cutoff values so the difference is much greater.

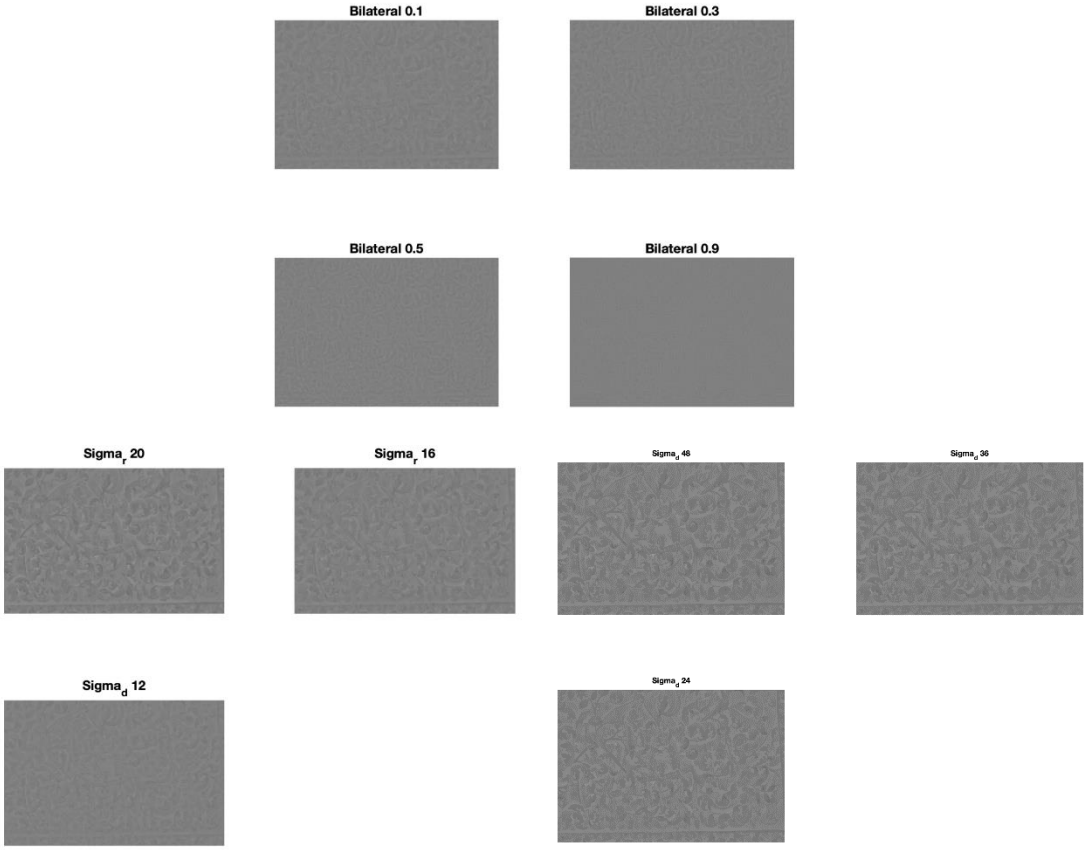


Fig. 11: Bilateral Filter Error Images

Final Error images are shown below run through identical parameters. Like with above, the lowest MSE in comparing our final images with the paper's final images was 4.5 in the case where the cutoff value = 0.9, $\sigma_r = 12$, $\sigma_d = 24$.

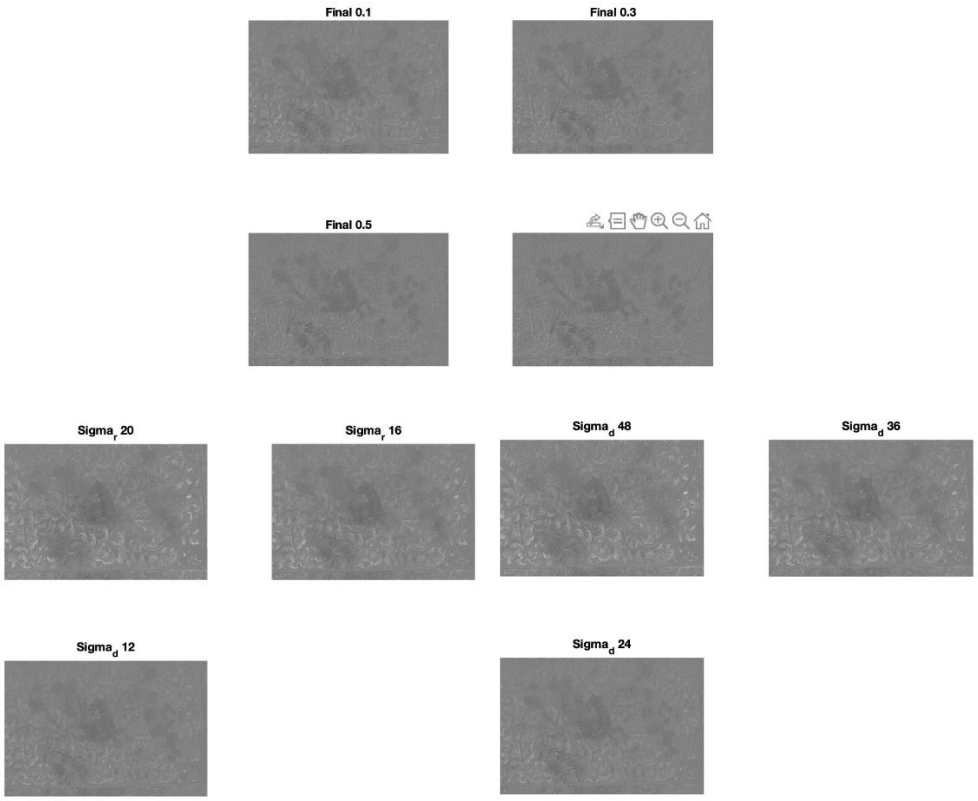


Fig. 12: Final Synthesis Error Images

5.5 Custom Image Pairs

We additionally ran our algorithm on new, custom image pairs intended to run the algorithm through its paces with areas of high and low detail, reflections, shadows, and color. Due to unforeseen complications with camera acquirement, the simple built-in camera of an iPhone7 was used to capture the scene. The four main elements are a detailed but monochromatic drawing of penguins, a detailed and colorful potted plant, a smooth and reflective ceramic bowl, and a detailed and reflective granite tabletop. The phone was propped up using two books out of frame as no flash and flash images were captured using default settings. As the scene and camera were static, liberties were taken with the amount of time between images on the order of seconds rather than fractions of a second. An additional method of lighting to flash image was also tried by simply turning on the room lights, which were aligned with the dimmer and warmer ambient light source to mitigate the occurrences of strange shadows.

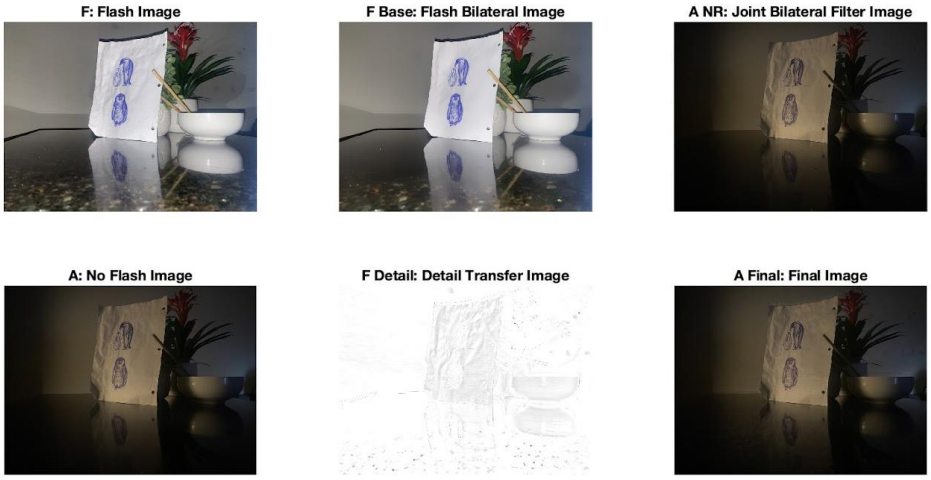


Fig. 13: Custom Image Pair Results, Standard Flash

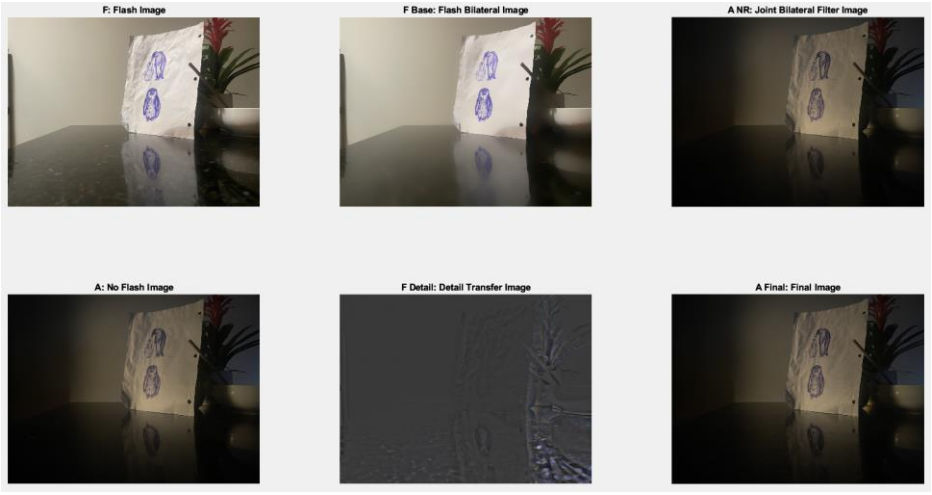


Fig. 14: Custom Image Pair Results, Room Light Flash

6 Red-Eye Correction Algorithm Analysis & Results

The red-eye artifact correction algorithm implemented in this project, with heavy inspiration from Petschnigg et al., can be summarized in the block diagram below:

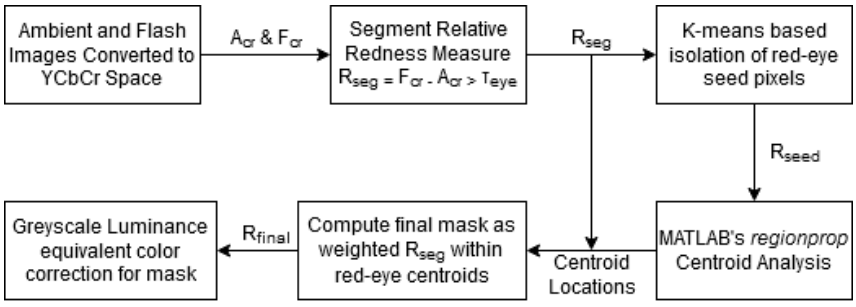


Fig. 15: Red-Eye Correction Algorithm Block Diagram

To perform red-eye correction, both the flash and ambient images are first converted into YCbCr space to allow for a direct comparison of red chrominances via what Petschnigg et al. described as a "relative redness measure" [2]. This primitive mask is simply the difference between the flash and ambient red chrominance channels, which is then segmented on a variable $\tau_{eye} = .05$ such that only substantially redder areas in the flash image are carried through. As seen on the sample image pair below, this first pass highlights several areas that are not the red eyes of interest.

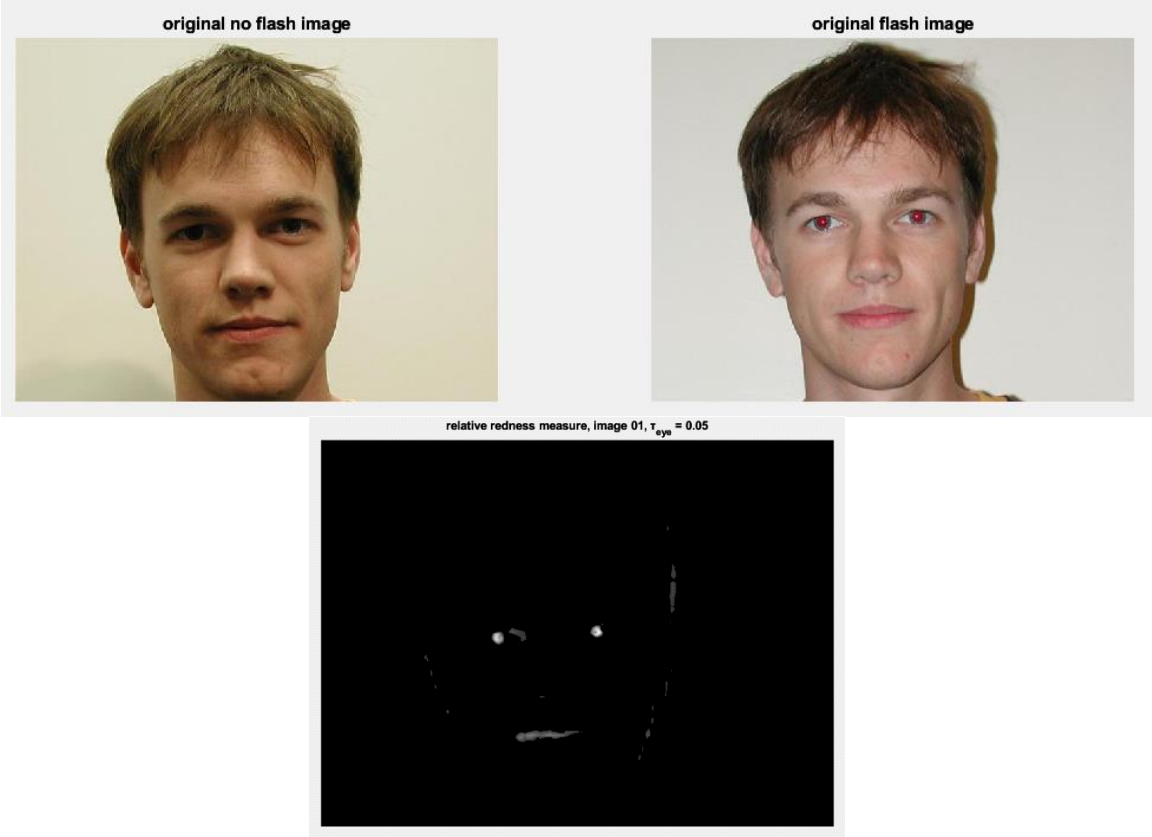
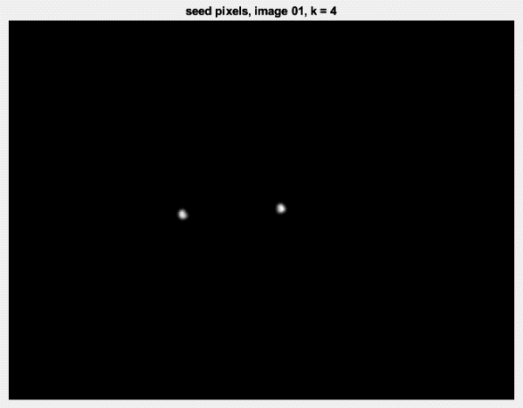


Fig. 16: Sample Pair & Resultant Simple-Threshold Relative Redness Measure Mask

In isolating the eyes for creating a mask, we differ from the Petschnigg et al. paper, opting to use MATLAB's various built-in functions. The image is further segmented using the *imsegkmeans* function which calculates the mean values of k clusters. For this project a k of 4 was used. As red-eyes form the highest areas of relative redness difference, the cluster with the highest mean was taken as red-eye candidate mask; after a simple gaussian blur to fully connect individual pupils, the mask has been improved as below:



Centroid		MajorAxisLength	MinorAxisLength
220.87	246.11	21.363	20.063
345.22	238.23	21.016	19.821

Fig. 17: K-Means Segmentation for Red-Eye Seed Pixels

Next MATLAB's *regionprops* function was used to find connected centroids with their minor and major axis lengths. Using the resultant ellipses, a final red-eye mask is formed from a feathered circular region of the simple-threshold relative redness measure mask. Lasty, specular highlights are found and removed from the mask using the same process described above for detail transfer leaving the final mask:

Fig. 18: Final Red-Eye Mask

The mask above is cropped to show as much detail as possible, including the specular highlights inside the pupils. This final mask is then applied in the same way as the Petschnigg et al. paper describes: each pixel in the mask is replaced with the greyscale equivalent of 80% of its ambient luminance.

Fig. 19: Final Red-Eye Removal Results

7 Conclusions

Stuff.

Appendix

The Matlab source code developed for this project so far can be found at the following address: https://github.com/jcw0525/ec520_flash

References

[1] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, Bombay, India, 1998, pp. 839-846, doi: 10.1109/ICCV.1998.710815.

[2] Georg Petschnigg, Richard Szeliski, Maneesh Agrawala, Michael Cohen, Hugues Hoppe, and Kentaro Toyama. 2004. Digital photography with flash and no-flash image pairs. ACM Trans. Graph. 23, 3 (August 2004), 664–672. https://doi.org/10.1145/1015706.101577

[3] A. K. Sinha, V. Bhateja, and A. Sharma, "Rayleigh based bilateral filtering for flash and no-flash image pairs," *2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN)*, Noida, India, 2015, pp. 559-563, doi: 10.1109/SPIN.2015.7095418.

[4] Elmar Eisemann and Frédo Durand. 2004. Flash photography enhancement via intrinsic relighting. ACM Trans. Graph. 32, 3 (August 2004), 673-678. https://doi.org/10.1145/1015706.1015778

[5] Hardeberg, Jon Yngve. “Red Eye Removal using Digital Color Image Processing.” *Image Processing, Image Quality, Image Capture Systems* Conference (2001).