BBM 444 – Programming Assignment 3: Flash/No Flash Photography

Due date: Monday, 22-04-2024, 23:59.

Overview

The purpose of this assignment is to explore photography with flash/no-flash pairs[1]. In general, getting the lighting in your scene right is one of the most important considerations one needs to take care of when taking a photograph. This can be particularly challenging when the scene you are trying to photograph has a large dynamic range, including both very dark and very bright objects. Previously, we saw a *passive* technique for photographing such difficult scenes, using HDR imaging. Here we will investigate an alternative *active* technique, where we insert new light into the scene using our camera's flash.

As we discussed in class, using flash has advantages and disadvantages. The extra light reduces noise, allows us to use larger shutter speeds to reduce motion blur, and can help illuminate the scene more uniformly. On the flip side, the use of the harsh flash lighting can ruin the scene's ambiance, produce highlights, and overall result in unappealing appearance. These issues commonly arise when using simple point flash lights, e.g., the one on a DLSR camera or at the back of a cell phone. A photographer with access to some more advanced equipment would use either a pointable flash that they can bounce off of the ceiling, or they would use a diffuse display to make the light smoother. However, for the purposes of this assignment, we can assume we do not have access to such equipment.

Instead, we can attempt to produce images that combine the advantages of flash and ambient illumination by capturing two images, one with each illumination, and then *fusing* them into one. In class we saw two types of techniques for this kind of *flash/no-flash photography*, one based on bilateral filtering, and another based on gradient-domain processing. In this assignment, you will implement both of these techniques, and apply them to photos provided by us, as well as photos you capture on your own.

Throughout the assignment, we refer to a number of key papers that were also discussed in class. Even though the assignment and class slides describe most of the steps you need to perform, we highly recommended that you read the associated papers. As always, there is a "Hints and Information" section at the end of this document that is likely to help.

Notation. Throughout this assignment we use bold font to denote image-sized quantities, and regular font to denote scalars. We use regular multiplication, division, squaring, absolute value, etc., notation involving image-sized quantities to denote element-wise operations. We use $\langle \cdot, \cdot \rangle$ to indicate the scalar resulting from the vector dot product between vectorized versions of the image-sized quantities.

1. Bilateral filtering (100 points)

In the first part of the assignment, you will use the lamp flash/no-flash images from the paper by Petschnigg et al. [1], available in the `./data/lamp` directory of the assignment ZIP archive. Your focus will be on *denoising* the ambient image using bilateral filtering. Figure 1 shows representative results for this section. As we discussed in class, bilateral filtering is a form of "blurring" that attempts to respect sharp edges in the image. This is achieved by weighting neighboring pixels not just based on their distance from the pixel that is being filtered, but also based on their similarity to it. This can help reduce noise in an image, without overly blurring details. Further improvements can be obtained by using bilateral filtering techniques that combine information from both the

[1] Adapted from the programming assignment developed by Ioannis Gkioulekas for his computational photography class.

Figure 1: From left to right: Photos taken under ambient lighting and flash lighting; denoising result using basic bilateral filtering; and final denoising result.

flash and ambient illumination images, as proposed by Petschnigg et al. [1]. Figure 3 of that paper is particularly helpful for navigating this assignment.

Implement bilateral filtering (40 points). As a first attempt at denoising, you can simply apply the bilateral filter on the ambient image. Using the notation of Petschnigg et al. [1], we will denote the ambient image $\boldsymbol{A}$ and the denoised image $\boldsymbol{A}^{\text{Base}}$. You will implement bilateral filter as described in the "PiecewiseBilateral" algorithm of Section 5 of the paper by Durand and Dorsey [2].

Implement joint-bilateral filtering (30 points). You will now attempt to get a better result by using joint bilateral filtering, as described in Section 4.1 of Petschnigg et al. [1], to produce the image $\boldsymbol{A}^{\text{NR}}$ in the paper's notation. As we discussed in class, joint bilateral filtering involves computing the intensity kernel using the flash image—see also Equation (4) of Petschnigg et al. [1].

Implement detail transfer (20 points). While joint bilateral filtering can help avoid overblurring the ambient image, it cannot *add* new detail to it. You will enhance the detail of your result by implementing the flash-to-ambient detail transfer procedure described in Section 4.2 of Petschnigg et al. [1]. For this, you will need to apply basic bilateral filtering to the flash image. Following Petschnigg et al. [1], we denote the flash image $\boldsymbol{F}$ and the result of filtering $\boldsymbol{F}^{\text{Base}}$. Then, you can form a new estimate for the ambient image by combining $\boldsymbol{F}$, $\boldsymbol{F}^{\text{Base}}$, and the image $\boldsymbol{A}^{\text{NR}}$ from before, as:

$$\boldsymbol{A}^{\text{Detail}} = \boldsymbol{A}^{\text{NR}} \frac{\boldsymbol{F} + \epsilon}{\boldsymbol{F}^{\text{Base}} + \epsilon} \tag{1}$$

where ϵ is some small value you can select on your own.

Implement shadow and specularity masking (10 points). Finally, it is time to create the mask $\boldsymbol{M}$ described in Section 4.3 of Petschnigg et al. [1]. This is used to detect regions in the flash image that have shadows or specularities not present in the ambient image. The mask can be generated using the simple thresholding operations described in the paper. Once you have the mask, form a final estimate $\boldsymbol{A}^{\text{Final}}$ of the ambient image as

$$\boldsymbol{A}^{\text{Final}} = (1 - \boldsymbol{M}) \cdot \boldsymbol{A}^{\text{Detail}} + \boldsymbol{M} \cdot \boldsymbol{A}^{\text{Base}} \tag{2}$$

Implementation details. These details apply to all four of the previous steps. As we discussed in class, the bilateral filter takes as input two parameters, the standard deviation σ_s of the spatial Gaussian kernel, and the standard deviation σ_r of the intensity Gaussian kernel. You will need to experiment with different values of these parameters to find which ones give the most aesthetically pleasing results. A good range to experiment with is $\sigma_s \in [1, 64]$ and $\sigma_r \in [0.05, 0.25]$. Note that these values assume that the maximum value in your image is 1, so make sure to normalize first.

In general, the best parameters will be different for each of the four types of filtering you will implement: basic bilateral filtering, joint bilateral filtering, denoising with detail transfer, and mask-based merging. In your report, you should compare the results of the four algorithms for different sets of parameter values. In these comparisons, you should make sure to show difference images $\text{image}_1 - \text{image}_2$, which should help highlight where in the images the techniques perform

Figure 2: Combining photographs taken under ambient lighting (left) and flash lighting (middle), to produce a new image that combines the best of both worlds (right).

differently. You should report the parameter values you chose as best for the four types of filtering, and discuss the advantages and disadvantages of each type.

2. Gradient-domain processing (100 points)

In the second part of the assignment, you will use the museum flash/no-flash images from the paper by Agrawal et al. [1], available in the `./data/museum` directory of the assignment ZIP archive. Figure 2(a-b) shows the two images. As you can see in the ambient image, the background painting is nicely illuminated from the ambient lighting. However, the person in the foreground is very dimly lit. We can attempt to address this using the camera's flash, which results in the flash photograph. Now the person in the foreground is well-illuminated, but our photograph suffers from a different problem: The sharp lighting of our flash has created a specular highlight on the painting. In this part of the assignment, you will focus on fusing the flash and ambient images, in order to produce uniform illumination without the specularity and other artifacts. Figure 2(c) shows the end result of the fusion. You will do this fusion using gradient-domain processing, as proposed by Agrawal et al. [3]. As usual, you are strongly encouraged to read through the paper, to find many implementation details and insights about the various parameters.

Differentiate and then re-integrate an image (75 points). This task will not give you any new images, but it should help you write and test the Poisson solver you will use next to create fused flash/no-flash images.

Take any image you want (e.g., the ambient image from the museum pair), which we will denote as $\boldsymbol{I}$. The gradient field $\nabla \boldsymbol{I}$ of this image will be a two-channel image, where one channel is the per-pixel partial-x derivatives $\boldsymbol{I}_x$, and the other channel is the per-pixel partial-y derivatives $\boldsymbol{I}_y$. You will attempt to integrate this gradient field to form a new image, $\boldsymbol{I}^*$, that should ideally be the same as the original image $\boldsymbol{I}$.

For this, you should first compute the divergence of the gradient field $\nabla \boldsymbol{I}$,

$$\operatorname{div}(\nabla \boldsymbol{I}) = \nabla \cdot \nabla \boldsymbol{I} = \boldsymbol{I}_{xx} + \boldsymbol{I}_{yy} \ , \tag{3}$$

which you can do by using Laplacian filtering on the original image $\boldsymbol{I}$. Then, you will use this divergence as input to your implementation of the Poisson solver. You will implement the conjugate gradient descent (CGD) algorithm we discussed in class. We recommend that you read through Sections 8 and B2 of Shewchuk's [4] excellent (and funny) discussion of CGD. Algorithm 1 shows the CGD procedure you should implement. In the algorithm, LaplacianFiltering is convolution with the Laplacian kernel, $\boldsymbol{B}$ is a binary image-sized mask that equals 0 on the boundary and 1 elsewhere, and $\boldsymbol{I}^*_{\text{boundary}}$ is an image equal at the boundary to the values used by the boundary conditions.

ALGORITHM 1: Gradient field integration with conjugate gradient descent.

Input: target divergence $\boldsymbol{D}$, initialization $\boldsymbol{I}^*_{\text{init}}$, boundary mask $\boldsymbol{B}$, boundary values $\boldsymbol{I}^*_{\text{boundary}}$, convergence parameter $\epsilon > 0$, iteration number N.
Output: integrated image $\boldsymbol{I}^*$.
/* Initialization */
$\boldsymbol{I}^* \leftarrow \boldsymbol{B} \cdot \boldsymbol{I}^*_{\text{init}} + (1 - \boldsymbol{B}) \cdot \boldsymbol{I}^*_{\text{boundary}}$;
$\boldsymbol{r} \leftarrow \boldsymbol{B} \cdot (\boldsymbol{D} - \text{LaplacianFiltering}\,(\boldsymbol{I}^*))$;
$\boldsymbol{d} \leftarrow \boldsymbol{r}$;
$\delta_{\text{new}} \leftarrow \langle \boldsymbol{r}, \boldsymbol{r} \rangle$;
$n \leftarrow 0$;
/* Conjugate gradient descent iteration */
while $\sqrt{\langle \boldsymbol{r}, \boldsymbol{r} \rangle} > \epsilon$ *and* $n < N$ **do**
  $\boldsymbol{q} \leftarrow \text{LaplacianFiltering}\,(\boldsymbol{d})$;
  $\eta \leftarrow \frac{\delta_{\text{new}}}{\langle \boldsymbol{d}, \boldsymbol{q} \rangle}$;
  $\boldsymbol{I}^* \leftarrow \boldsymbol{I}^* + \boldsymbol{B} \cdot (\eta \cdot \boldsymbol{d})$;
  $\boldsymbol{r} \leftarrow \boldsymbol{B} \cdot (\boldsymbol{r} - \eta \cdot \boldsymbol{q})$;
  $\delta_{\text{old}} \leftarrow \delta_{\text{new}}$;
  $\delta_{\text{new}} \leftarrow \langle \boldsymbol{r}, \boldsymbol{r} \rangle$;
  $\beta \leftarrow \frac{\delta_{\text{new}}}{\delta_{\text{old}}}$;
  $\boldsymbol{d} \leftarrow \boldsymbol{r} + \beta \cdot \boldsymbol{d}$;
  $n \leftarrow n + 1$;
end

For initialization, set $\boldsymbol{I}^*_{\text{init}}$ equal to the zero image. For Dirichlet boundary conditions, set the outermost pixels of $\boldsymbol{I}^*_{\text{boundary}}$, at all four image edges, to equal the corresponding pixels of the original image $\boldsymbol{I}$. You can experiment with different values for the parameters ϵ and N controlling the convergence of your algorithm. If your implementation is correct, then the final image $\boldsymbol{I}^*$ should be equal to the original image $\boldsymbol{I}$.

Create the fused gradient field (25 points). In this part, we will follow the notation of Agrawal et al. [3], to make it easier to compare with the paper. Note that some quantities reuse symbols from the bilateral filtering question, but *are not necessarily the same.* We will denote the ambient image as $\boldsymbol{a}$, the flash image as $\boldsymbol{\Phi}'$, and the fused image as $\boldsymbol{\Phi}^*$. The first step in the fusion process requires forming the gradient field that will be used to create $\boldsymbol{\Phi}^*$. In turn, this gradient field will be created using the gradient fields $\nabla \boldsymbol{a}$ and $\nabla \boldsymbol{\Phi}'$ of the ambient and flash image, respectively.

First, compute $\nabla \boldsymbol{a}$ and $\nabla \boldsymbol{\Phi}'$. As with $\nabla \boldsymbol{I}$, these are two-channel images, with one channel corresponding to partial-x and the other partial-y derivatives. Then, use these gradients to implement the following steps:

1. Compute the *gradient orientation coherency map* $\boldsymbol{M}$ using Equation (5) of Agrawal et al. [3],

$$\boldsymbol{M} = \frac{|\nabla \boldsymbol{\Phi}' \cdot \nabla \boldsymbol{a}|}{||\nabla \boldsymbol{\Phi}'||\,||\nabla \boldsymbol{a}||} = \frac{|\boldsymbol{\Phi}'_x \cdot \boldsymbol{a}_x + \boldsymbol{\Phi}'_y \cdot \boldsymbol{a}_y|}{\sqrt{\boldsymbol{\Phi}'^{\,2}_x + \boldsymbol{\Phi}'^{\,2}_y}\sqrt{\boldsymbol{a}_x^2 + \boldsymbol{a}_y^2}} \,. \tag{4}$$

   Note that, in this equation, both the numerator and denominator are computed pixel-wise, and the mask $\boldsymbol{M}$ is image-sized.

2. Compute the pixel-wise *saturation weight map* $\boldsymbol{w}_s$ from the flash image $\boldsymbol{\Phi}'$ using Equation (11) of Agrawal et al. [3],

$$\boldsymbol{w}_s = \tanh\left(\sigma \cdot (\boldsymbol{\Phi}' - \tau_s)\right) \,. \tag{5}$$

   Normalize the saturation map to be in the range [0, 1].

3. Compute a new gradient field (two-channel image) for the fused image $\boldsymbol{\Phi}^*$ using Equation (12) of Agrawal et al. [3],

$$\nabla\boldsymbol{\Phi}^* = \boldsymbol{w}_s \cdot \nabla\boldsymbol{a} + (1 - \boldsymbol{w}_s)(\boldsymbol{M} \cdot \nabla\boldsymbol{\Phi}' + (1 - \boldsymbol{M}) \cdot \nabla\boldsymbol{a}) \,, \tag{6}$$

or equivalently,

$$\begin{bmatrix} \boldsymbol{\Phi}^*_x \\ \boldsymbol{\Phi}^*_y \end{bmatrix} = \begin{bmatrix} \boldsymbol{w}_s \cdot \boldsymbol{a}_x + (1 - \boldsymbol{w}_s)(\boldsymbol{M} \cdot \boldsymbol{\Phi}'_x + (1 - \boldsymbol{M}) \cdot \boldsymbol{a}_x) \\ \boldsymbol{w}_s \cdot \boldsymbol{a}_y + (1 - \boldsymbol{w}_s)(\boldsymbol{M} \cdot \boldsymbol{\Phi}'_y + (1 - \boldsymbol{M}) \cdot \boldsymbol{a}_y) \end{bmatrix} \,. \tag{7}$$

You should experiment with different values for the scalar parameters σ and τ_s, starting with the values recommended in the paper. In your report, you should show the gradient fields $\nabla\boldsymbol{a}$, $\nabla\boldsymbol{\Phi}'$, and $\nabla\boldsymbol{\Phi}^*$, and mention the parameter values you used.

Finally, integrate the gradient field $\nabla\boldsymbol{\Phi}^*$, by computing its divergence and providing it as input to the Poisson solver you implemented earlier. The result will be a final fused image $\boldsymbol{\Phi}^*$. We recommend using Dirichlet boundary conditions: Set the outermost pixels of $\boldsymbol{\Phi}^*$, at all four image edges, to equal the corresponding pixels of either the ambient image, or the flash image, or their average. Likewise, you can use different initializations, corresponding to either the ambient image, or the flash image, or the average of the two, or even the all-zero image. Experiment with different boundary conditions and initializations, and show the final fused image you obtain.

3. Capture your own flash/no-flash pairs (100 points)

Now it is time to apply what you implemented above to your own pictures. You should capture two flash/no- flash pairs. The first pair should be suitable for applying the denoising techniques based on bilateral filtering. Good examples include dimly lit environments, e.g., a room illuminated by just a desk lamp, or an outdoors scene at night illuminated by only a street lamp. You can look at Petschnigg et al. [1] for inspiration.

The second pair should be suitable for applying the fusion algorithm based on gradient-domain processing. To create results which are clearly better than any single exposure, you should take pictures of a scene where flash affects some parts of the image, while leaving others relatively unaffected. Good examples include dark scenes that have both matte and specular objects. You can look at Agrawal et al. [3] for inspiration.

The total number of points you will get for this part will depend on how visually compelling the final fused images you create are.

What to Hand In

Your submitted solution should include the following:

- The filled-in Jupyter Notebook as both your source code and report. The notebook should include (1) markdown cells reporting your written answers alongside any relevant figures and images and (2) well-commented code cells with reproducible results.
- Image files for Problems 1 and 2, showing the various fusion results you are asked to generate.
- Image files that you create in Problem 3, showing the original flash and no-flash pairs you capture with your camera, and the fused images you generate from them. You can also include additional image files for various experiments (e.g., fusion with different values) other than the final ones if you think they show something important.

You should prepare a ZIP file named name-surname(s)-assgn3.zip containing the files stated above and, and submit it via email to `abtasdemir@cs.hacettepe.edu.tr`.

When submitting your work, please follow these guidelines:

- Use the subject line: "BBM 446 Assignment 3"
- Include your full name and student ID in the email body.
- Ensure that the file you share is accessible.

Late policy

You may use up to five *extension* days (in total) over the course of the semester for the programming assignments. Late submission will not be allowed.

Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your other classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered as a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.

Hints and Information

- When dealing with color images, you should apply all the algorithms you implement to each color channel separately. (Though note the hint below about the mask $\boldsymbol{M}$.)
- In your bilateral filtering implementation, you do not need to implement Gaussian filtering on your own. Instead, you can use OpenCV's `gaussianblur` function.
- The "PiecewiseBilateral" algorithm of Durand and Dorsey [2] includes an InterpolationWeight step, which the paper implements using linear interpolation. We recommend implementing this as follows: First, accumulate the images J^j for all j values in a NB_SEGMENTS-channel image, where the j-th channel equals J^j. Then, use `scipy`'s `interpolate.interpn` to interpolate the channels and form a final single-channel image.

  Additionally, when creating the segments, you should make sure that the first value i^j is smaller than the smallest value in your image (typically 0), and the last value i^j is larger than the largest value in your images (typically 1). You can do this by defining $\text{minI} \equiv \min(I) - \lambda$ and $\text{maxI} \equiv \max(I) + \lambda$, where λ is a small constant (e.g., $\lambda = 0.01$, should be smaller than σ_r). Then, use these adjusted values of minI and maxI to set $\text{NB\_SEGMENTS} = \text{ceil}\left(\frac{\text{maxI}-\text{minI}}{\sigma_r}\right)$ and $i^j = \text{minI} + j \cdot \frac{\text{maxI}-\text{minI}}{\text{NB\_SEGMENTS}}$.
- When debugging your bilateral filter implementation of the bilateral filter, it can be helpful to compare your output with that of OpenCV's `bilateralFilter`. However, you should *not* use the OpenCV implementation to produce any results.

- When computing the mask $\boldsymbol{M}$ for the shadow and specularity masking in Equation (2), you will get better results by using *linear* images. However, the images you are provided with are non-linear, and you do not have an exposure stack you can use to perform radiometric calibration. As a default, you can apply the gamma correction operator of the *sRGB standard* [5]. This is the inverse of the gamma encoding operator you used in assignments 1 and 2, and corresponds to the following transformation:

$$C_{\text{linear}} = \begin{cases} \frac{C_{\text{non-linear}}}{12.92}, & C_{\text{non-linear}} \leq 0.0404482, \\ \left(\frac{C_{\text{non-linear}}+0.055}{1.055}\right)^{2.4}, & C_{\text{non-linear}} > 0.0404482, \end{cases} \tag{8}$$

  Additionally, the mask $\boldsymbol{M}$ may turn out to have a lot of small disconnected areas, or large connected areas with small gaps. These can result in strong artifacts in your final image $\boldsymbol{A}^{\text{Final}}$. You can get better results by using *morphological filtering* operations (dilation, erosion, opening, and closing) to post-process the individual shadow and specularity masks, before combining them to form the mask $\boldsymbol{M}$. Finally, note that even though you should apply the various bilateral filtering operations separately on each channel, you will likely get better results by computing a mask $\boldsymbol{M}$ from the luminance of the ambient and flash images, and using this same mask for all color channels. You can experiment with all these options and see what gives you better results.

- To solve this assignment, you will need to implement the gradient, divergence, and Laplacian differential operators. We strongly recommend creating three subroutines, `gradient`, `divergence`, `laplacian`, that you reuse throughout the assignment whenever you need to compute these quantities. To compute the first-order derivatives in the gradient and divergence operators, we recommend using `numpy`'s `diff` function *with appropriate padding*. To compute the Laplacian, we recommend using `scipy`'s `signal.convolve2d` function to convolve with the Laplacian kernel,

$$\Delta = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \tag{9}$$

  with arguments `mode='full', boundary='fill', fillvalue=0`.

  The gradient and divergence implementations require different padding; figuring out exactly how to pad can be tricky and will require some trial-and-error. To test your implementation, you should ensure that for any arbitrary image $\boldsymbol{I}$, the following holds

$$\texttt{divergence}(\texttt{gradient}(\boldsymbol{I})) = \texttt{Laplacian}(\boldsymbol{I}) \tag{10}$$

  at all pixels *except* at the boundary (first and last row, first and last column). Note that the incorrect boundary values is the reason we use the mask $\boldsymbol{B}$ when computing the residual in Algorithm 1.

- The algorithms in this assignment can work with both RAW and rendered (e.g., PNG or JPEG) images. Typically, working with RAW will give you better looking results, but in that case you will also need to use `dcraw` and tonemapping. We leave it up to you to decide what images to use.

References

[1] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama. Digital photography with flash and no-flash image pairs. *ACM Transactions on Graphics (TOG)*, 23(3):664–672, 2004.

[2] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 257–266, New York, NY, USA, 2002. ACM.

[3] A. Agrawal, R. Raskar, S. K. Nayar, and Y. Li. Removing photography artifacts using gradient projection and flash-exposure sampling. *ACM Transactions on Graphics (TOG)*, 24(3):828–835, 2005.

[4] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. School of Computer Science. *Carnegie Mellon University, Pittsburgh*, PA, 15213:10, 1994.

[5] International Electrotechnical Commission and others. IEC 61966-2-1:1999. *Multimedia systems and equipment–Colour measurements and management–Part 2-1: Colour management–Default RGB colour space–sRGB*, 1999.