Homework Assignment 3

15-463/663/862, Computational Photography, Fall 2020
Carnegie Mellon University

Due Monday, Oct. 19, at 11:59pm

The purpose of this assignment is to explore photography with flash/no-flash pairs. 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 the class DLSR or at the back of your 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 classes of techniques for this kind of *flash/no-flash photography*, one based on bilateral filtering, and another based on gradient-domain processing. In this homework, 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. While the assignment and class slides describe most of the steps you need to perform, it is 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. Thoughout this assignment we use bold font to denote image-sized quantities, and regular font to denote scalars. We use regular multiplication, division, squaring, 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)

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

In the first part of the homework, you will use the lamp flash/no-flash images from the paper by Petschnigg et al. [6], available in the `./data/lamp` directory of the homework 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 flash and ambient illumination images, as proposed by Petschnigg et al. [6]. Figure 3 of that paper is particularly helpful for navigating this homework.

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. [6], 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 paper by Durand and Dorsey [3]. We leave it up to you to decide whether to implement the vanilla algorithm, or any of the efficient algorithms described in Section 5 of that paper. (The "PiecewiseBilateral" algorithm is generally about as complicated to implement as the vanilla algorithm, and significantly faster; the "FastBilateral" algorithm can be tricky to implement due to the downsampling and upsampling steps.)

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. [6], 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. [6].

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. [6]. For this, you will need to apply basic bilateral filtering to the flash image. Following Petschnigg et al. [6], 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. [6]. 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 differently. You should report the parameter values you chose as best for the four types of filtering, and iscuss the advantages and disadvantages of each type.

2. Gradient-domain processing (100 points)

In the second part of the homework, 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 homework 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

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).

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 homework, you will focus on fusing the flash and ambient images, in order to produce uniform illumination without the specularity and other artifacts. The end result of the fusion is shown in Figure 2(c). You will do this fusion using gradient-domain processing, as proposed by Agrawal et al. [1]. 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 $\boldsymbol{I}$. Compute its gradient field $\nabla \boldsymbol{I}$—this 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 now attempt to integrate this gradient field to form a new image, which we will denote as $\boldsymbol{I}^*$. For this, you should first compute the divergence,

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

Then, you should implement the Poisson solver based on conjugate gradient descent (CGD) we discussed in class. For this, we recommend that you read through Sections 8 and B2 of Shewchuk's [7] 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, and EnforceBoundaryConditions is the procedure we discussed in class for enforcing Dirichlet boundary conditions.

For initialization, set $\boldsymbol{I}^{*,0}$ equal to the zero image. For Dirichlet boundary conditions, set the outermost pixels of $\boldsymbol{I}^*$, 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 controling 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. [1], 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. [1],

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

ALGORITHM 1: Gradient field integration with conjugate gradient descent.

Input: target divergence $\operatorname{div}(\nabla \boldsymbol{I})$, initialization $\boldsymbol{I}^*_{\text{init}}$, boundary conditions $\boldsymbol{B}$, convergence parameter $\epsilon > 0$, iteration number N.

Output: integrated image $\boldsymbol{I}^*$.

/* Initialization */

$\boldsymbol{I}^* \leftarrow \text{EnforceBoundaryConditions}(\boldsymbol{I}^*_{\text{init}}, \boldsymbol{B})$;

$\boldsymbol{r} \leftarrow \operatorname{div}(\nabla \boldsymbol{I}) - \text{LaplacianFiltering}(\boldsymbol{I}^*)$;

$\boldsymbol{d} \leftarrow \boldsymbol{r}$;

$\delta_{\text{new}} \leftarrow \langle \boldsymbol{r}, \boldsymbol{r} \rangle$;

$n \leftarrow 0$;

/* Conjugate gradient descent iteration */

while $\|\boldsymbol{r}\| > \epsilon$ *and* $n < N$ **do**

  $\boldsymbol{q} \leftarrow \text{LaplacianFiltering}(\boldsymbol{d})$;

  $\eta \leftarrow \frac{\delta_{\text{new}}}{\langle \boldsymbol{d}, \boldsymbol{q} \rangle}$;

  $\boldsymbol{I}^* \leftarrow \text{EnforceBoundaryConditions}(\boldsymbol{I}^* + \eta \cdot \boldsymbol{d}, \boldsymbol{B})$;

  $\boldsymbol{r} \leftarrow \boldsymbol{r} - \eta \cdot \boldsymbol{q}$;

  $\delta_{\text{old}} \leftarrow \delta_{\text{new}}$;

  $\delta_{\text{new}} \leftarrow \langle \boldsymbol{r}, \boldsymbol{r} \rangle$;

  $\beta \leftarrow \frac{\delta_{\text{new}}}{\delta_{\text{old}}}$;

  $\boldsymbol{d} \leftarrow \boldsymbol{r} + \beta \cdot \boldsymbol{d}$;

  $n \leftarrow n + 1$;

end

2. Compute the pixel-wise *saturation weight map* $\boldsymbol{w}_s$ from the flash image $\boldsymbol{\Phi}'$ using Equation (11) of Agrawal et al. [1],

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

   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. [1],

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

   or equivalently,

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

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 $\nabla \boldsymbol{\Phi}^*$ to form the final fused image $\boldsymbol{\Phi}^*$, using the Poisson solver you implemented earlier. 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. [6] 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

ALGORITHM 2: Gradient field integration with preconditioned conjugate gradient descent.

Input: target divergence $\text{div}\,(\nabla \boldsymbol{I})$, initialization $\boldsymbol{I}^*_{\text{init}}$, boundary conditions $\boldsymbol{B}$, preconditioner $\boldsymbol{P}$, convergence parameter $\epsilon > 0$, iteration number N.

Output: integrated image $\boldsymbol{I}^*$.

/* Initialization */

$\boldsymbol{I}^* \leftarrow \text{EnforceBoundaryConditions}\,(\boldsymbol{I}^*_{\text{init}}, \boldsymbol{B})$;

$\boldsymbol{r} \leftarrow \text{div}\,(\nabla \boldsymbol{I}) - \text{LaplacianFiltering}\,(\boldsymbol{I}^*)$;

$\boldsymbol{d} \leftarrow \boldsymbol{P}^{-1} \cdot \boldsymbol{r}$;

$\delta_{\text{new}} \leftarrow \langle \boldsymbol{r}, \boldsymbol{d} \rangle$;

$n \leftarrow 0$;

/* Preconditioned conjugate gradient descent iteration */

while $\|\boldsymbol{r}\| > \epsilon$ *and* $n < N$ **do**

  $\boldsymbol{q} \leftarrow \text{LaplacianFiltering}\,(\boldsymbol{d})$;

  $\eta \leftarrow \frac{\delta_{\text{new}}}{\langle \boldsymbol{d}, \boldsymbol{q} \rangle}$;

  $\boldsymbol{I}^* \leftarrow \text{EnforceBoundaryConditions}\,(\boldsymbol{I}^* + \eta \cdot \boldsymbol{d}, \boldsymbol{B})$;

  $\boldsymbol{r} \leftarrow \boldsymbol{r} - \eta \cdot \boldsymbol{q}$;

  $\boldsymbol{s} \leftarrow \boldsymbol{P}^{-1} \cdot \boldsymbol{r}$;

  $\delta_{\text{old}} \leftarrow \delta_{\text{new}}$;

  $\delta_{\text{new}} \leftarrow \langle \boldsymbol{r}, \boldsymbol{s} \rangle$;

  $\beta \leftarrow \frac{\delta_{\text{new}}}{\delta_{\text{old}}}$;

  $\boldsymbol{d} \leftarrow \boldsymbol{s} + \beta \cdot \boldsymbol{d}$;

  $n \leftarrow n + 1$;

end

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. [1] 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.

4. Bonus: Reflection removal in flash/no-flash photography (50 points)

A different potential problem when using flash is that it can create reflections off of glass surfaces. See Figure 6 in Agrawal et al. [1]. Implement the fusion algorithm described in Section 4.1 ("Observation Deck") of the paper, and then use it on a flash/no-flash pair you capture with your own camera. We suggest using a scene with a window for demonstrating this part, as in the paper.

5. Bonus: Implement acceleration techniques for Poisson solver (50 points)

We mentioned in class two techniques for accelerating Poisson solvers based on the CGD method. Implement simple versions of both techniques:

Implement Jacobi preconditioning (25 points). Read Sections 12 and B3 of Shewchuk [7], and implement the preconditioned conjugate gradient descent (PCGD) algorithm, shown in Algorithm 2. Given that the matrix $\boldsymbol{L}$ corresponding to Laplacian filtering is diagonally-dominant, you can use the so-called *Jacobi preconditioner* $\boldsymbol{P} = \text{diag}\,(\boldsymbol{L})$, which is one of the simplest but quite effective preconditioners. Note that PCGD requires computing matrix-vector products involving he inverse of the preconditioner. You should make sure to take advantage of the special structure of the Jacobi preconiditioner $\boldsymbol{P}$ to compute these matrix-vector products efficiently, without forming the inverse matrix.

Implement a simple multi-grid approach (25 points). Multi-grid algorithms require going back-and-forth between different resolutions while running the CGD algorithm. Instead of implementing such an algorithm, you will implement a much simpler heuristic approach: First, form Gaussian pyramids of all the images you use as input to your Poisson solver. Then, run your Poisson solver at progressively higher resolutions, starting at the smallest resolution of your Gaussian pyramids and ending at the highest

resolution (which should match the resolution of the original images). Each time you increase resolution, use as initialization an up-sampled version of the solution you obtained at the previous resolution.

Evaluation. Apply both acceleration techniques, as well as their combination, on the integration tasks you implemented in Problem 2. Discuss differences between the baseline CGD algorithm and the accelerated versions in terms of convergence speed and final results.

6. Bonus: Other types of gradient-domain processing (up to 50 points)

Gradient-domain processing is a general image processing paradigm that can be used for a broad set of applications, including blending, tone-mapping, colorization, converting to grayscale, edge enhancement, image abstraction and non-photorealistic rendering, and so on.

GradientShop [2] is a paper and associated app that provide a unifying framework for expressing all these tasks as gradient-domain processing algorithms. Read this paper, and implement whichever of the applications described within appeal to you the most (the instructor's personal favorite is non-photorealistic rendering, but it won't reflect badly on your grade if you go for something else). For additional inspiration, you can take a look at the results in the Poisson image editing paper [5].

Your implementation does not need to follow GradientShop exactly. For instance, instead of implementing the "long-edge detector" described in the paper, you can come up with your own detector of continuous edges. We also recommend *not* trying the video-based applications.

In your submitted report, you should make sure to describe in detail exactly what you did. This includes what task you chose to implement, where you followed and where you deviated from GradientShop, and in the latter case what you did differently. You should also show at least three examples showcasing the effects of your gradient-domain processing pipeline.

Deliverables

As mentioned on Piazza and the course announcement, solutions for assignment 3 are submitted through Gradescope. Your submission should include the following files:

- A PDF report explaining what you did for each problem, including answers to all questions asked throughout Problems 1 and 3, as well as any of the bonus problems you choose to do. The report should include any figures and intermediate results that you think may help. Make sure to include explanations of any issues that you may have run into that prevented you from fully solving the assignment, as this will help us determine partial credit. The report should also explain any additional image files you include in your solution (see below).
- All of your Python code, including code for the bonus problems, as well as a `README` file explaining how to use the code.
- 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 your final ones, if you think they show something important.
- If you do Bonus Problem 4: The original flash/no-flash images you captured and the corresponding fused image files you produced.
- If you do Bonus Problem 6: The original and corresponding processed image files you used.

Hints and Information

- 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* [4]. This is the inverse of the gamma encoding operator you used in homework assignmens 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 $\boldsymbol{M}$.

- Whenever you compute image gradients in this homework, you should do it by either convolving with the forward-difference kernels,

$$\begin{bmatrix} 1 & -1 \end{bmatrix}, \text{ and } \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \tag{9}$$

or using Numpy's `diff` function. Do *not* use the function `gradient`, as it implements central differences, which will result in inconsistencies with the Laplacian kernel used for the Poisson solver. You should use the same functions when computing divergence. Whenever you need to perform Laplacian filering in this homework, you should do it by convolving with the 2D Laplacian kernel,

$$\boldsymbol{\Delta} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{10}$$

- During conjugate gradient descent, make sure to only compute the divergence $\nabla \cdot \nabla \boldsymbol{\Phi}^*$ once, instead of recomputing it in every iteration.
- When performing convolutions in the gradient-domain processing part of the homework, you should be careful about the various padding arguments `scipy.signal.convolve2d` provides. Given the type of boundary conditions you are using, you should always do convolution with the `mode='same'` argument.
- When dealing with color images, you should apply all of the algorithms you implement to each color channel separately.
- The algorithms in this homework 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.

Credits

Some inspiration for this assignment and the write-up came from Oliver Cossairt's computational photography course at Northwestern University. The lamp and museum flash/no-flash pairs came from Petschnigg et al. [6] and Agrawal et al. [1], respectively.

References

[1] 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.

[2] P. Bhat, C. L. Zitnick, M. Cohen, and B. Curless. Gradientshop: A gradient-domain optimization framework for image and video filtering. *ACM Transactions on Graphics (TOG)*, 29(2):10, 2010.

[3] 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.

[4] 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.

[5] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. In *ACM Transactions on graphics (TOG)*, volume 22, pages 313–318. ACM, 2003.

[6] 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.

[7] 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.