CSCI 3290 Computational Photography

Assignment 2 – Image Blending

Due Date: **11:59pm on Friday, November 1st, 2013**

1. Backgrounds

The primary goal of this assignment is to seamlessly blend an object or texture from a source image into a target image. The simplest method would be to just copy and paste the pixels from one image directly into the other. Unfortunately, this will create very noticeable seams, even if the backgrounds are similar. How can we get rid of these seams without doing too much perceptual damage to the source region?

In this project, you are required to implement two well-known blending algorithms in MATLAB: the **Laplacian pyramid blending** and **Poisson blending**. Both algorithms focus on the gradient of the input images.

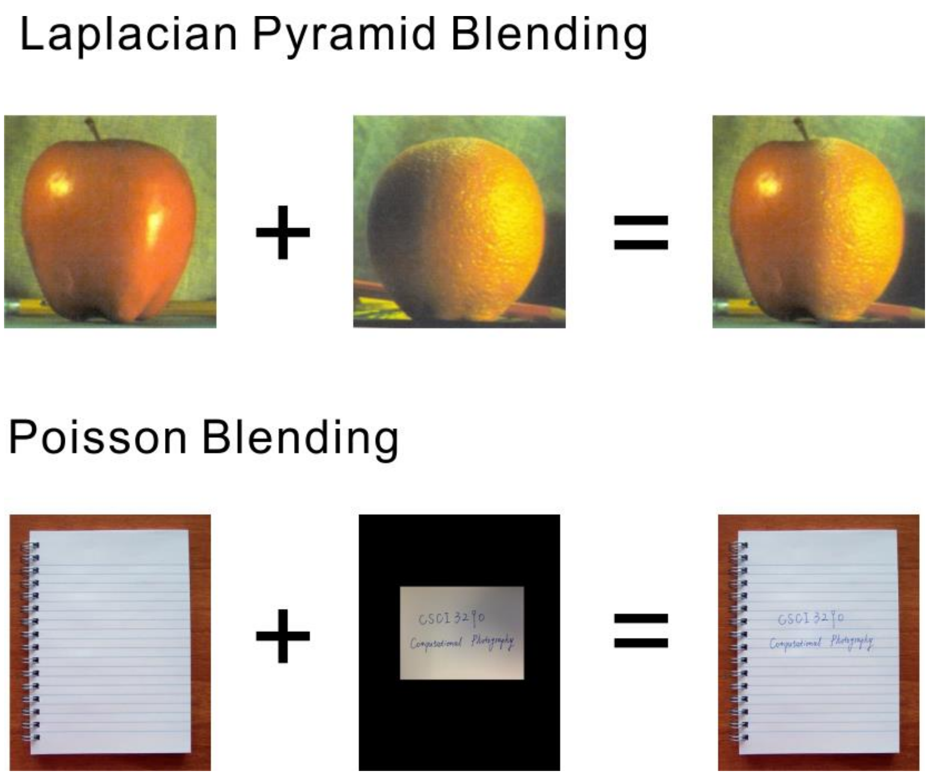


Figure 1. Demonstration of Blending Algorithms

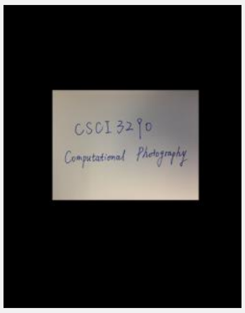
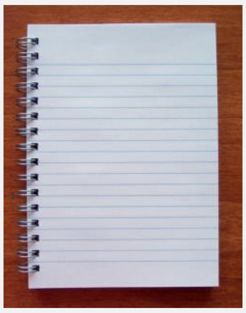
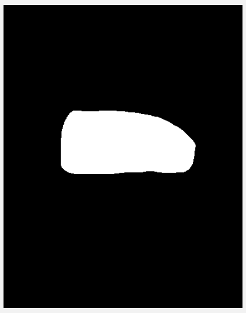
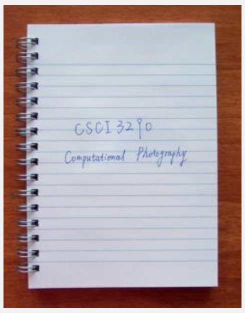
Laplacian pyramid blending is introduced in Chapter 6. The first step of this algorithm is to construct two Laplacian pyramids for both input images. Instead of combine the original images directly, we combine their Laplacian pyramid and reconstruct the result based on the pyramid. The detail steps will be shown later.

Poisson blending **solves a linear system of equations**, such that the difference between the intensity on boundary between the background and foreground is as small as possible. Also, Poisson blending tries to minimize the gradient difference across pixels between the background and foreground in their overlapped region. In Figure 1, you can find how powerful the Poisson blending is.

2. Problem

For both blending algorithm, your input are always the following three images:

1. Foreground image
2. Background image
3. Mask

Foreground	Background	Mask	Output
			

The task of blending algorithm is to generate a synthetic image which looks natural and contains content of both foreground and background images. Different blending algorithms generate different results. What you need to do is to implement **BOTH** two algorithms: Laplacian pyramid blending and Poisson blending.

3. Algorithms & Implementation Details

Laplacian Pyramid Blending

The detail of this algorithm is introduced in your lecture notes Chapter 6. To finish your assignment, you can use our skeleton and follow several steps (or you can also write your own code and follow your own logic to complete this blending algorithm.):

1. Write a module to generate Image/Laplacian/Gaussian pyramids for an input image.
2. Use the module in step 1 to generate Image/Laplacian/Gaussian pyramids for the foreground, background and mask separately. (some of the pyramids may not be useful, so you can treat them as byproducts)
3. Use the Laplacian pyramids of the foreground and background, and the Gaussian pyramids of the mask to generate a mixed Laplacian pyramids.
4. Use the smallest scale images of the foreground and the background, and the smallest scale Gaussian of the mask to generate a mixed small scale image.
5. Use the mixed small scale image and the mixed Laplacian pyramids gained from step 3&4 to reconstruct a new image with two image details blended.

We will give you more implementation hints later.

Poisson Blending

The detail and solver of this algorithm are introduced in your lecture notes Chapter 7 and tutorial notes 5. The basic idea of the algorithm is like this:

Assuming that our input image is a 400x300 single channel image. There are 120000 pixels in total. What we want to solve is a result image which we represent as a 120000x1 vector **x**. Then we construct an equation **Ax**=**b** which defines what condition the vector **x** should fulfill, i.e. how the result image looks like. **A** is a Rx120000 matrix and **b** is a Rx1 vector where R means there are R equations in this linear system. You should **design these R equations** in order to obtain the desired result by solving this linear system **Ax=b**. But how to design the **R** equations so as to form the matrix **A** and vector **b**? Hint: you may consider the pixel value/gradient of the foreground and background images.

You can use our skeleton and following several steps to finish your work (or you can also write your own code and follow your own logic to complete this blending algorithm.):

1. Construct a sparse matrix A (R equations' left hand side) with the MATLAB call `sparse(i,j,s,m,n)`
2. Construct vector B (R equations' right hand side)
3. Solve the linear system with MATLAB operation `A\B`

4. Extract and output the final result from **x**

We will give you more implementation hints later.

4. TODO List

We provide you the following files as the skeleton code:

In folder “pyramid/”: **blendingPyramid.m**, **createPyramid.m**

In folder “poisson/”: **blendingPoisson.m**

(Don't be afraid. If your program is well organized, it will be no more than 200 lines of code in total)

We also provide some test images:

In folder “pyramid/”: **apple.png, apple2.png, apple_mask.png**

In folder “poisson/” (2 sets):

notebook.png, notebook2.png, notebook_mask.png;

snow.png, snow2.png, snow_mask.png.

You need to add your own code to complete the skeleton program we provided (or you can also write your own code to achieve the same target with your own logic):

TODO 1~4: (Laplacian pyramid blending)

TODO #1. Write a module to generate Image/Laplacian/Gaussian pyramids for an input image. (See **createPyramid.m**)

TODO #2. Combine the Laplacian pyramids of the background and foreground. (See **blendingPyramid.m**)

TODO #3. Combine the smallest scale image of the background and foreground. (See **blendingPyramid.m**)

TODO #4. Reconstruct the image with the results gained from TODO #2 and TODO #3. (See **blendingPyramid.m**)

TODO 5~6: (Poisson blending)

TODO #5. Construct matrix A and B based on the pixel value and gradient of the foreground and background images. (See **blendingPoisson.m**)

TODO #6. Extract the final result from R (R=A\B) and output the result. (See **blendingPoisson.m**)

5. Important Hints

To give you more guidance for finishing your assignment, we provide you some hints about the implementation detail (please read this part carefully):

1. In TODO #1, the image pyramid can be generated by using `imresize`. If the size of image is [H, W], the next level in the pyramid should be [H/2, W/2].
2. In TODO #1, the Gaussian pyramid can be approximated by using `imresize` to resize the image from scale [H,W] to [H/2, W/2], and then resize it back from [H/2, W/2] to [H,W]. After this

procedure, the image will be blurred and suitable for generating the Laplacian of an image.

3. In TODO #1, the Laplacian of an image can be approximated by the formula **L = I – G**, where **I** is the original image, and **G** is its Gaussian.
4. In TODO #2 and TODO #3, to combine 2 images with a mask, you can do it with simple operation in MATLAB: **mask.*foreground + (1-mask).*background**.
5. In TODO #4, you can loop the following procedure from smallest scale to the original scale: use `imresize` to upsample the image from Level **K** to Level **K-1**, and add the Laplacian of Level **K-1** to the image
6. In TODO #5, since we treat an image as a 1D vector, every pixel will be thought as an index instead of a coordinate **(x,y)**. If your **k**-th equation is expressing: pixel **p** – pixel **q** = value **v**, you should set 2 values in sparse matrix A and 1 value in vector B, i.e. put 1 into **(k, p)** cell of matrix A, put -1 into **(k, q)** cell of matrix A, put **v** into the **k**-th cell of vector B.
7. In TODO #5, we suggest you consider the pixel value of the background in the region outside the mask, and consider the sum of the gradient of the background and foreground in the region inside the mask. (For more detail, you can Google "mix gradient problem")
8. In TODO #6, since the result is a 1D vector, you should rearrange its data to the form of an image.

6. Marking

The upper bound score of this assignment is 120.

Basic Part (100%)

(40%) The implementation of Laplacian pyramid blending.

(40%) The implementation of Poisson blending.

(20%) Report (include results, brief introductions and comparison of 2 algorithms, your own examples, execution time etc.)

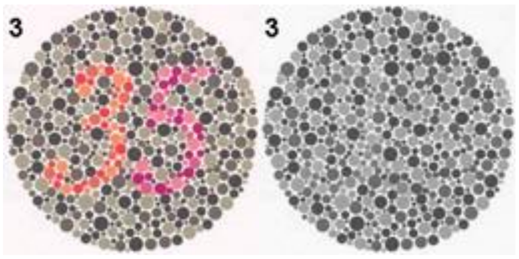
Extra Credit (20%)

The upper bound of extra credit is 20 percent. Be sure to demonstrate your bonus part clearly in your report. You can consider the following ideas:

Bonus 1 (4%): Demonstrate some failure cases for BOTH 2 blending algorithms. Tell us why it fails.

Bonus 2 (6%): For Poisson blending, solve **Ax=b** with other techniques, such as gradient descent and conjugate gradient methods.

Bonus 3 (10%): Implement a Color2Gray application with the technique you learned from Poisson blending. Color2Gray: Sometimes, in converting a color image to grayscale (e.g., when printing to a laser printer), we lose the important contrast information, making the image difficult to understand. For example, compare the color version of the image on right with its grayscale version produced by rgb2gray(). Can you do better than rgb2gray? Gradient-domain processing provides one avenue: create a gray image that has similar intensity to the rgb2gray output but has similar contrast to the original RGB image. This is an example of a tone-mapping problem, conceptually similar to that of converting HDR images to RGB displays. To get credit for this, show the grayscale image that you produce (the numbers should be easily readable). Hint: Try converting the image to HSV space and looking at the gradients in each channel. Then, approach it as a mixed gradients problem where you also want to preserve the grayscale intensity.



7. Submission

Your submission should contain two things: **code files (in MATLAB)** and **a report (in HTML)**

MATLAB Code

1. The completed code files (or your own implementation): **blendingPyramid.m**, **createPyramid.m**, **blendingPoisson.m**
2. Any other code you would like to add.

Report Requirements

1. The report must be written in **HTML**.
2. In the report, you should describe your algorithm and any decisions you made to write your algorithm a particular way. You should also show and discuss the results of your algorithm.
3. Discussion on algorithms' efficiency (based on time elapsed).
4. Discuss any extra credit you did.
5. Ideas and algorithms from others **MUST** be clearly claimed. List papers or links in the **Reference** section if needed.
6. Feel free to add any other information you feel is relevant.

Submission Format

The folder you hand in must contain the following:

1. **README.txt -** containing anything about the project that you want to tell the TAs, including brief introduction of the usage of the codes
2. **code/ -** directory containing all your code for this assignment.
3. **html/ -** directory containing all your html report for this assignment (including images). Your web page should only display compressed images (e.g. jpg or png).
4. **html/index.html -** home page for your results

Please compressed the folder into <*your student ID*>-asgn2**.zip** or <*your student ID*>-asgn2**.rar**, and upload it to e-Learning system.

8. Remarks

1. Five late days total, to be spent wisely.
2. 20% off per day for late submission.
3. Your mark will be deducted if you do not follow the instructions for the submission format.
4. The three assignments are to be completed **INDIVIDUALLY** on your own.
5. You are encouraged to use methods in related academic papers.
6. Absolutely **NO sharing or copying** of code! **NO sharing or copying** of reports! Offender will be given a failure grade and the case will be reported to the faculty.