Coursework Assignment C - Group 12
CS4125 - Seminar Research Methodology For Data Science

Liliana Oliveira – 4767306
Nivedita Prasad – 4712099
Aishwarya Shastry – 4743016

April 8, 2018

Contents

1 Algorithms Implemented

1.1 TASK 1: Gradient-based Image Sharpening

The Algorithm (Image sharpening)
Input: (Pixel values of input image),c_s,c_u (parameter)
STEP 1 Construct gradient matrix G
STEP 2 Compute gradients of input image $\overline{g} = G\overline{U}$
STEP 3 Solve $\min_X(\| GU - c_s\overline{g} \|^2 + \| c_u(U - \overline{U} \|^2)$
STEP 4 Return minimizer U

1.2 TASK 2: Gradient-based Image Blending

Algorithm for Gradient Image Blending
Input Source and Target Images
Step 1

- Create vector U for storing inner and boundary pixels of the source
- Create a vector U^b which stores boundary pixels of the target

Step 2 Calculate Gradient matrix G that contains only the inner and boundary gradients.
Step 3 Calculate selector matrix S that selects the boundary pixels
Step 4 Calculate gradients of source image and create a vector g which stores the source gradients and 0 for boundary gradients.

2 Functionality of the Implementation

2.1 Image Sharpening

The algorithm mentioned in section 1.1 is repeated for all the 3 channels that is " d " (RED,GREEN and the BLUE) for every pair of neighboring pixel. **Construction of Gradient matrix G :** The *GRADIENT matrix G* of m rows and n columns maps the *image U* to the *gradient vector g*. Here, gradient matrix G is computed using the user-defined *function G*. This matrix G is stored as a Sparse matrix as the output of the function G. The input to the function G are the variables *h,w* where h is the height of the image and w is the width of the image. Further, inside the function G we initialize three variables *i,j* and *v* which are the input to the sparse matrix. The input i is the row index and input j is the column index, whereas the input v is the value indexed by the combination of i,j. These variables are initialized to zeros using the " zeros " function in matlab and made into a column vector. The size of all the 3 vectors is the summation of 2*h*(w-1) and 2*w*(h-1). As the next and the most important step is to fill up i,j with indexes and v with values. For this we decided to do the following: The index sequence we decided for i in matlab is as follows : we stacked

$$1 : w * h - 1; 1 : w * h - 1; 1 : h * w - h; 1 : h * w - h$$

on top of each other which then was formed into a column vector .Then the index sequence we decided for j in matlab is as follows: we stacked

$$1 : w * h - 1; 2 : w * h; 1 : h * w - h; h + 1 : h * w$$

on top of each other which was formed into a column vector. This stacking done to accommodate the values of v with the ones function in of matlab like follows :

$$0.5 * ones(w * h - 1, 1); -0.5 * ones(w * h - 1, 1); -0.5 * ones(h * w - h, 1); 0.5 * ones(h * w - h, 1)$$

The filled up i,j and v is given as input to sparse matrix which is the finally computed **G** .

The above algorithm is used to *construct* the gradient matrix G that is made up of difference in the gradients of the pixels considered in two directions namely *right to left and up to down.* The number of g vectors g_k to be found depends on h and w of the image U. The image gradients are calculated where the difference in neighboring pixel values U_i and U_j is found on the whole image U. The gradient vector g_k is found using the below method:

$$g_k = 0.5 * (U_j - U_i)$$

Figure 1: Comparison of Sharpened and Blurred Image

As the next step, the image to be sharpened is read and then reshaped. The value of c_u, c_s is hard-coded to 0.5 and 3.0 respectively. Then gradient function is called to compute gradient matrix G. The Gradient matrix G is filled with 0.5, -0.5 and 0 in planned manner such that when G is multiplied with image U a gradient vector g is formed as a product which is repeated " d " times. Now as the last step, the minimizer function is computed to produce the final SHARPENED image. This output image is reshaped before it is displayed. Figure 1 shows the final result of the TASK 1's algorithm. For a clear understanding, the matlab functions "gradient.m" and "myTask.m" that is attached with this report can be executed.

2.2 Image Blending

The algorithm in section 1.2 uses a binary mask to select the image or a portion of source image to be put on target image. We begin with adding an extra boundary of 1 pixel around the masked image which will be removed after blending. Gradient matrix G is calculated for inner gradients and boundary gradients.The input to the function G are the variables *h,w* where h is the height of the image and w is the width of the image.We further form the matrix to define the boundary in the target.

Gradient vector g is calculated for each channel. We further ask the user where to put the masked image. getpts takes the x and y coordinates and puts the masked image there. The selector matrix has the values from all the corners and with width and height we calculate the i, j vectors in the perspective of the target. We further calculate a sparse matrix with (i,J) and value of the pixel calculated by the formula where m is the total number of rows in

the picture, i and j would be the row and column.

$$value = (j-1)m + i$$

Ub stores the boundary pixels of the target. We take a greater than 0 and we would try different values of a to compute U to see the impact of a on the resultant image in the below formula:

$$(Gt' * Gt + a * S' * S)U = Gt' * gb + a * S' * Ub$$

We unfortunately did not get the output due to dimension of matrices incompatible.

Algorithm 1.2 when implemented properly would result in the images as below:

Figure 2: Source Image

Figure 3: Target Image

Figure 4: Blended Image

We can use a freehand to select the area we want to copy to the target. It would look like the image below:

Figure 5: Freehand

3 Tests for Correctness of the Implemented Algorithms

Test of correctness is done for different input images and the results obtained are evaluated. The Task 1 is tested on a Test image and the resultant image is displayed in figure 6 .

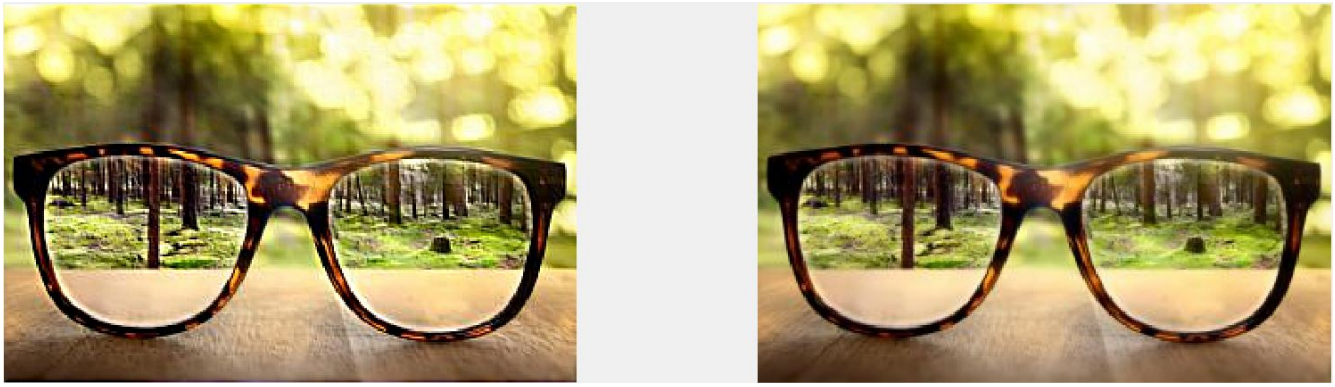

Figure 6: Test - Comparison of Sharpened and Blurred Image

Fig6 shows that the sharpening has been done on some parts of the image. The image on the left side is the sharpened image and the original image is on the right side.

4 Benefits and Limitations of the method for the applications

The algorithm provided for Task 1 is to compute gradients,re-scale them and construct a new (sharper) image whose gradients best match the rescaled gradients which may be limitation as the image can be sharpened more. Actual implementation contains the first and second order differentiation of the image which helps in Laplacian filtering which is further scaled and enhanced or sharpened with addition to the original image. The benefit of this process of sharpening helps in understanding the picture clearly without any ambiguity.

Algorithm for Task 2 takes source, target and mask image as input, calculates inner and boundary gradients for source image, boundary gradients for the target image, resizes image from the source and constructs an image which blends masked image to a specified location in target image. Gradient Blending preserves the gradients so that the masked image is blended into the target image smoothly .Major limitation of this technique is that,it takes only rectangular masked images. Another limitation is the images need to be properly aligned. Color of the the source image is easily influenced by the color of the surrounding target image pixels, which may result in highly undesirable effect. This could be solved by using Laplacian Pyramind.

5 Division of labor amongst the group members

Task 1 (Image Sharpening) done by Nivedita Prasad (4712099) and
Task 2 done by Aishwarya Shastry (4743016) and Liliana Oliveira (4767306)