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

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

June 15, 2018

Contents

1 Task 1: Gradient-based Image Sharpening

The blurry source image ”blurryImage.png” is an RGB image of size 354 x 204 x 3 ; i.e 216648 bytes and each value is an 8 bit integer.

1.1 Algorithms Implemented

General Explanation : The algorithm is given an input image that has to be sharpened, and 2 parameters c_s and c_u. The c_s controls the scaling of the gradients and c_u penalizes the deviation from the input image. The output resulting from this algorithm is an sharpened image.

Steps of Algorithm:

1. The blurred image U is first read and is reshaped into height x width rows and 3 RGB color (d) columns. Then the value of c_s is set to 3.5 and the value of c_u is set to 0.5.

2. The next step is to calculate gradient matrix G. For this, we calculate by using height h and width w of the image that has to be sharpened and returns a Gradient matrix G which is sparse. This Gradient matrix G will have 3 parameters i,j,v. The i th row and j th column of the gradient matrix will be filled with values v that is either positive 0.5 or negative 0.5 . First we calculate the horizontal gradient and then the vertical gradient.

3. Zero column vectors are created using ”zeros” function and is to index variables for i, j, v. The number of zeros in the column vector is determined by the formula 2*h*(w-1)+2*w*(h-1). Then we calculate the *m* which is the total of number of horizontal and vertical gradients to be calculated.

4. Horizontal gradient and Vertical gradient are calculated in the gradient function and the sparse matrix is filled with 0.5 and -0.5.

5. For all 3 dimensions, we calculate the optimization function , which minimizes the

   $$G' * G + c_u * I * d)U = c_s * G' * g + c_u * U$$

   This penalizes the deviation and sharpens the input image U. Hence the Gradient based Image Sharpening is achieved. The files gradient_g.m and myTaskSharpening.m has our implemented algorithm.

1.2 Use of Implemented Tool

The user has to provide a *blurry image* that has to be sharpened which is U and we extract height, width and dimension for gradient calculation. Then the *output* image is generated after the *scaling and penalization parameter* is also set. The crickter image that is blurred is sharpened now.

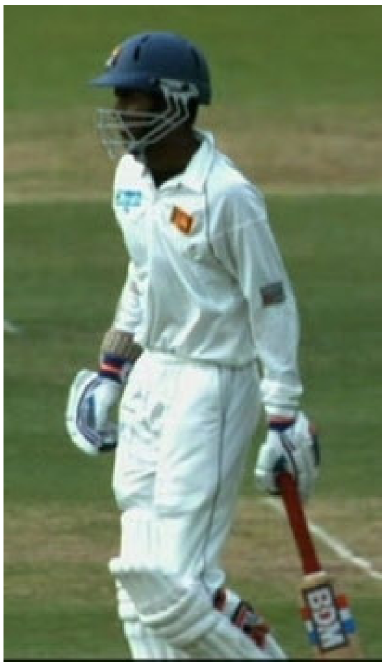


Figure 1: Blurry Cricketer

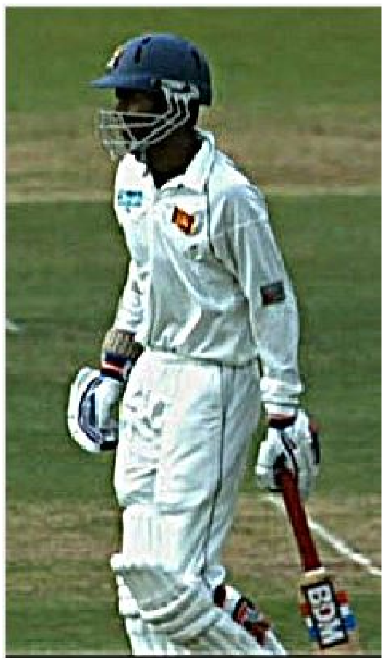


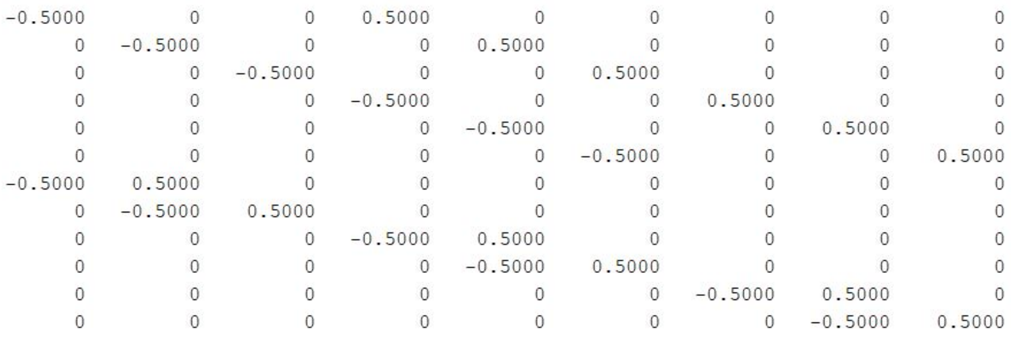
Figure 2: Sharpened Cricketer

1.3 Functionality Implementation Offers

The sharpening tool we have created using matlab, helps in simplifying ”implementation’ and ”Testing” as per lecture slides. Hence the tool will sharpen the given input image along with the aid of the parameters c_s that the scales the gradients and c_u penalizes the deviation from the input image which can be set too.

1.4 Tests for Correctness of the Implemented Algorithms

We first implemented the 3 x 1 pixel image given to us in the lecture to understand the concept of gradient g and then scaled it to a larger input image. Hence, to check the correctness of our gradient calculation g, we implemented gradient_test_sharpening_final.m which takes a 3 x 3 image as input and checks if the gradient calculated by hard-coding compared to our gradient function gradient_g.m gives the same value. The final output Gradient Matrix G, of the gradient_test_sharpening_final.m *before* applying sparse() function looks like Fig 3. Then when the image U multiplied by G gives gradient g_test which is same as g_calculated, which is tested with the method *isequal* that returns 1 if the comparison is equal/true and 0 otherwise. We obtained value 1 hence the correctness test is executed and verified. We have converted the sparse matrix G back to normal full matrix G only for display purpose as in Figure 3.



```
   -0.5000          0          0     0.5000          0          0          0          0          0
         0    -0.5000          0          0     0.5000          0          0          0          0
         0          0    -0.5000          0          0     0.5000          0          0          0
         0          0          0    -0.5000          0          0     0.5000          0          0
         0          0          0          0    -0.5000          0          0     0.5000          0
         0          0          0          0          0    -0.5000          0          0     0.5000
   -0.5000     0.5000          0          0          0          0          0          0          0
         0    -0.5000     0.5000          0          0          0          0          0          0
         0          0          0    -0.5000     0.5000          0          0          0          0
         0          0          0          0    -0.5000     0.5000          0          0          0
         0          0          0          0          0          0    -0.5000     0.5000          0
         0          0          0          0          0          0          0    -0.5000     0.5000
```

Figure 3: Gradient G

1.5 Evaluation of resulting Tool

To evaluate , we used different forest image to check whether the image get sharpened as well. For example, we use as input the blurry picture shown in Figure 4 and as a result we get a sharpen image as shown in Figure 5. Hence the tool is working fine.

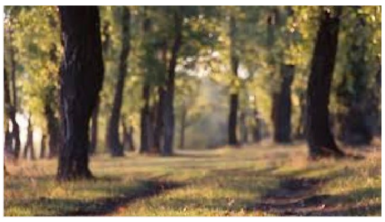


Figure 4: Blurry Forest Image

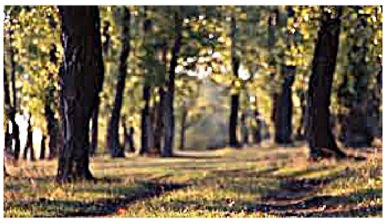


Figure 5: Sharpened Forest Image

1.6 Benefits and Limitations of method implemented

Benefits: The user has to just provide the blurred image and set the c_s and c_u to sharpen the image.
Our algorithm works for most of the images and produce good results for sharpening as seen in Fig2 and Fig5.
Limitations : The current method has limitations, if the images are both blurred and shaky as can be seen from Fig 6 and Fig7. For our current task, we have set c_s and c_u value to 3.5 and 0.5 respectively. On changing these values, we can get better results. These values can be controlled by end user for better results.

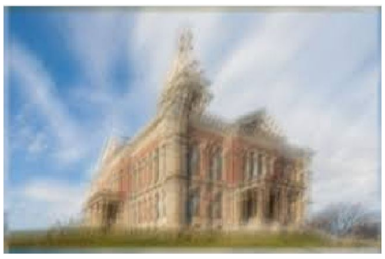


Figure 6: Blurry Building image

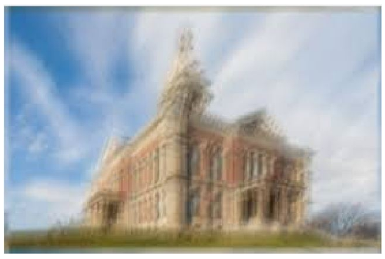


Figure 7: Sharpened Building Image

2 Task 2: Gradient-based Image Blending

2.1 Algorithm Implemented

General Explanation : The algorithm is given one or more source images that has to be blended with one target image, and the selection of the regions are assumed to be rectangles and of same shape. The output resulting from this algorithm an blended image of the source and target.

Steps of Algorithm:

1. The Source Image and Target Image is read using im_read and the we get their height, width and dimension and then we calculate the inner gradient(gradient_g) for the source image.
2. Now we assign to U_s the whole region of the target that has the dimensions of the source and we substitute the inner pixels by the source image. Hence we have a vector where the boundary is from the target and the inner pixels are from the source.
3. Then we compute the gradient matrix G that contains only the inner gradients and the boundary gradients.
4. Next we compute the selector matrix S , that selects the boundary pixels U_b . Now we compute the gradients of the source image (g) and change it in order to store the boundary gradients as zero. (We could only achieve left blending, but the same procedure could be applied to the top, right and bottom edges).
5. To obtain the blended image we compute the formula
$$G' * G + alpha * S' * S) * U = G' * g + alpha * S' * U_b$$
6. Finally, we just paste the blended image in the target image in the coordinates (s,e) defined in the beginning of the implementation.

2.2 Use of Implemented Tool

This tool can be used to merge two images into one with blending. For this task, the user has to provide a source and a target image, which would be then analyzed and a single image would be given as output. The user has to just give the images to be blended and the tool developed using matlab blends them. As seen in the figure 8 and 9, we could achieve the blending of the left boundary alone.

Figure 8: Source Image

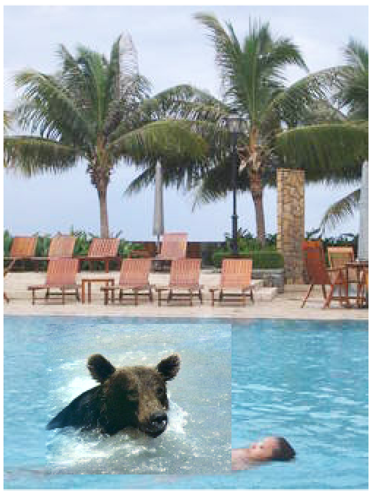


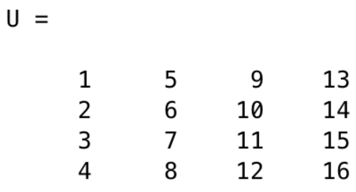
Figure 9: Blended Image

2.3 Functionality Implementation Offers

The Matlab code allows users to have a source and target image of their choice and blend the two images at a position they would like to put the image in which is variable s and e in our matlab code (coordinates of the position where the source has to be copied into the target image). For our initial implementation, we have given s and e the values 220 and 30 and to alpha values as 1 which can be changed as per user needs.

2.4 Tests for Correctness of the Implemented Algorithm

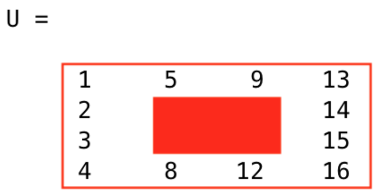
We implemented a test for the selector matrix S, which performed in *testSelectorMastrix.m*, by giving it a 4x4 matrix and we wanted to check if the output of our implementation matched the expectations. In the code if the vectors assigned to the sparse matrix match the expected results for both vectors, the method prints "true", else it prints "Not matching".



```
U =

     1     5     9    13
     2     6    10    14
     3     7    11    15
     4     8    12    16
```

Figure 10: Original Matrix



```
U =

     1     5     9    13
     2                14
     3                15
     4     8    12    16
```

Figure 11: Expected Values

For the blending performance we tested for another image and , as expected, it also worked for the left boundary alone, as shown in Figure 12

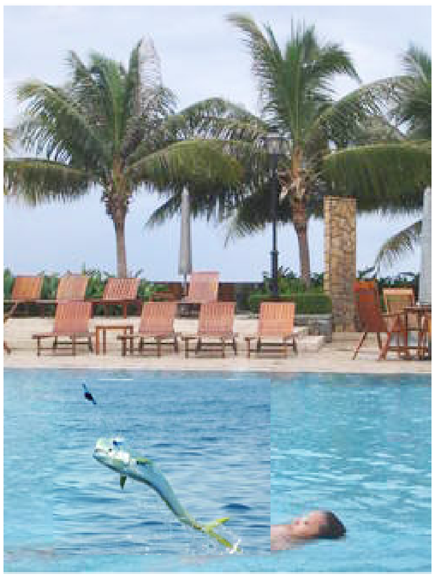


Figure 12: Blending with other image

2.5 Evaluation of resulting Tool

Image blending is used to mix the boundary of the source image and the boundary of the target position to have the same color so that we will have the illusion that the source actually belongs to the target image and not just pasted there. Visually the left boundary is clearly blended with the target image.

2.6 Benefits and Limitations of method implemented

Limitations: We think if the color of the boundaries is almost same of the source and target image, then the blended image will have a fine transition of colors. This is where the tool may fail. Otherwise as provided in the exercise the color blue was present in both the source and the target image boundaries will show a clean transition. We could also have asked the user to specify the position by showing him the target image and he just needed to specify the point by clicking on it. *Benefits:* The user can blend images and do a blending effect similar to photo-shop without the help of such expensive tools or prior user experience of such tools. For future enhancements, to improve the method, one can use a freehand mask which would result in higher quality results.

3 Division of labor amongst the group members

Task 1: Gradient-based Image Sharpening: by Nivedita Prasad (4712099) and Liliana Oliveira (4767306)
Task 2: Gradient-based Image Blending: by Nivedita Prasad (4712099), Liliana Oliveira (4767306) and Aishwarya Shastry (4743016)