Lab 2

September 12, 2018

1 Exercise 1: Fade Two Images and Merge Them

1.1 Version 1:

- You are going to use `Cameraman_image` already loaded in the Part 1. Load the second image `5.1.09.tiff` which is a photo of the surface of the moon. Assign this image to the variable `Moon_image`. Following this, show `Moon_image`.

```
In [ ]: figure(1)
        imshow(Moon_image);
```

- Fade `Moon_image` by a factor of 0.8 and `Cameraman_image` by factor of 0.2 and sum the two matrices in new matrix named `Mixte_image`. Show `Mixte_image` below.

```
In [ ]: figure(2)
        imshow(Mixte_image);
```

- Create a new image named `First_part_image_1` by selecting the first 100 components of each dimension of image Cameraman_image.
- Create a new image named `last_part_image_2` by selecting the last 100 components of each dimension of image Moon_image.
- Fade both images, `last_part_image_2` by a factor of 0.8 and `First_part_image_1` by factor of 0.2 and sum the two matrices in new matrix named `last_part_Mixte_image`.

1.2 Version 2:

- We will do the same fade and mixing process as version 1 question but this time using matrix and vector multiplication.
- The first step will consist of changing both image matrices from size of 256x256 each to vector of size 56536x1; (To create a vector from matrix uses the function `reshape`. Use the help to show you how to use `reshape` function.
- Create a new matrix (named `Both_images`) by appending both image vectors to form a matrix of size 56536x2.
- Create a vector named `Fade_vector` of size 2x1 and containing the fade factor values (0.5,0.5) for both images.
- Multiply `Both_images` matrix and `Fade_vector` to obtain the mixing_image_vector.

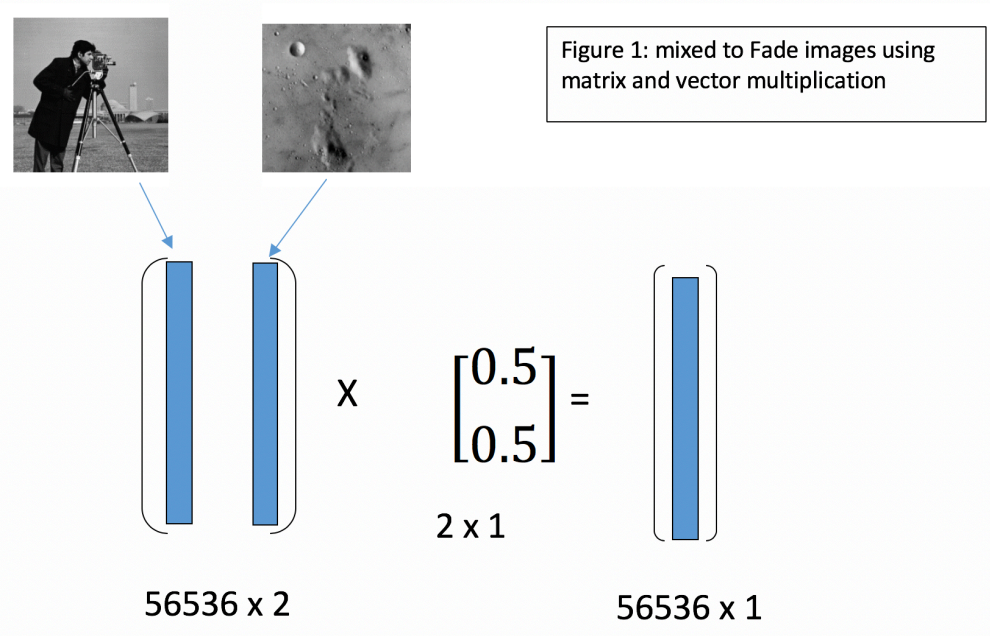

image.png

- Resize the obtained vector `mixing_image_vector` using the function `reshape` to create `mixing_image_matrix` of size (256x256).

2 Exercise 2:

In this exercise, you will create a demo for Newton's method graphically. Specifically, you need to create a video of how Newton's method progresses with iterations. Every step and command to use is mentioned in the comments in `Lab2_NewtonMethod.m` file. The polynomial function is given by

$$f(x) = x^3 - 7x + 1$$

The polynomial and its derivative should be hardcoded in the files `poly.m` and `poly_derivative.m`. You can load the polynomial using `f = @poly`. Value of the function at any value, say 0, can be found using `f(0)`. You can load the derivative using `fder =@poly_derivative`. Value of the derivative at any value, say -1, can be found using `fder(-1)`. Test your code(demo) with different initial points. Note: You are not limited to use same mentioned commands.

2.0.1 For Python:

- You can use the `numpy` library.
- `numpy.polynomial.polynomial.polyval` can be used to calc the function at any value.
- `numpy.polynomial.polynomial.polyder` can be used to calc the derivative at any value.
- You can plot the animation by `matplotlib` (examples here or using `plotly`.