**Project 2**

**Maryam Darei Computational Photography**

**Dr. Pless**

***Hybrid Images***

**Background**

A hybrid image is an image that is perceived in one of two different ways, depending on viewing distance, based on the way humans process visual input. These images are composite images that blend two pictures--one picture is filtered using a high pass, and the other is filtered with a low pass. When the two filtered images are layered over each other, the human eye will see one or the other, depending on the viewing distance.

**Data Description:**

Data Type: Both RGB and grayscale images of any resolution are valid inputs

**Data Characteristics:**

* We have two RGB images as an input which would have the potential to hybrid with each other
* Attribute breakdown: 2 quantitative input variables, and 1 quantitative output variable
* Missing Attribute Values: None
* The output after filtering should be the same size as the input image.

**Sample of input and output:**

|  |  |
| --- | --- |
| Input 1 | Input 2 |
| A cat looking at the camera  Description automatically generated | A dog with its tongue out  Description automatically generated with medium confidence |

**Data pre-processing and Feature Engineering:**

* 1. **Match the image’s size:** We should match two input images’ sizes before hybrid and create output.
  2. **Alignment images:** The alignment of these images is very important for a better final result. If we want to use the faces photos, we can align them based on eyes location. It means we should their eyes are in the same place.
  3. **Horizontal and vertical transitions:** The difference in the spatial position of the color channels (B, G, and R) causes a slip in the process of overlapping the three channels to form an RGB image, and we have color shadows in the produced images. In other words, the photo will not be of good quality. To solve this problem, the coordinate positions of all three channels must be exactly the same.

**Filters characteristic:**

We should have two filters: a lowpass filter and a high pass filter. Be careful that the image filter must be an odd-number width and height, to have an explicit center pixel.

For the low pass filter, we can use a standard 2D Gaussian filter or Blur filter and we make the high pass filter by subtracting the low pass filter from the original image.

A simple blur can be done using this kernel:

A picture containing text, clock

Description automatically generated

And below matrix is the kernel for the Gaussian Blur:

Calendar

Description automatically generated

**Code description:**

I have written it in **MATLAB**.

In the first step, after reading the input image, I divide each image into 3 parts horizontally in the main body of the program.

Graphical user interface, text, application, chat or text message

Description automatically generated

I developed a function to merge these 3 channels to each other and produce an RGB image. I developed it in two different approaches (“**MergetoRGB”, and “MergetoRGB2**”) that are very similar.

In the first one, based on the Professor’s advice, I assumed the Green as a reference and cropped a small window of the red and blue channel, for example, a 20X20, 50X50, or 100X100 in the middle of each red and blue channel. It is parametric and you should set it as input when you call the function. For each red and blue channel, I use a two-nested “For” loop and shift the selected parts to reach the best position. The amount of the shift ((e.g. [-5,5], [-15,15], or [20,20] pixels) is another input that you should set when you call the function. To acquire the best position, I use the Sum Squared Differences as a score and decide the best amount of the shift based on this value. In the first approach, I use the “circshift” for a shift matrix.

In the second approach, everything is the same as a first approach, but I made a little difference. Instead of using the “circshift” for a shift an image matrix, I use a new copy of the blue and red channel based on “i” and “j (the amount of search window in every step of the two-nested “For” loop). It means, I did not use a “circshift”. The reason of using this method is because with using of the “circshift”, in every shift step we replace the left amount with the rightest amount and indeed produce extra value.

I run these two functions and the program with a different value for search window length and different dimensions cropped of the image’s channels and compare the result.

**First Approach (“MergetoRGB”, d is the cropped window length, a is the search window length):**

**Result Analyze:**

It seems the second approach is better than the first one. It is completely clear that in the first approach with increasing the amount of “a” (the Search window in for loops) we could mess in the RGB image check the resulting numbers 2, 3, 4, 7, and 8 in the first approach). For the first approach, the best result is when we choose d=50 and a=20 (It is good to mention again that a is the dimension of the search window and d is the dimension of the cropped image).

In the second approach, we have a better result and less mess in the RGB images. In this case, the best results are for the d= 50 and a=20 or a=50.

**Result summary table:**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **D = 20** | **D = 50** |
|  | **a=2** |  |  |
| First | **a=20** |  |  |
| approach | **a=50** |  |  |
|  | **a=100** |  |  |
|  | **a=2** |  |  |
| Second | **a=20** |  |  |
| approach | **a=50** |  |  |
|  | **a=100** |  |  |

**Run Time:**

If you check the run time in 16 cases, you can understand the run time is increased with a bigger cropped image and with the bigger shift window.

**Bells & Whistles (Extra Points):**

One of the things that can be done in this concept is displaying different emotions in faces. One of the high or low spatial frequencies corresponds to faces with ”sad” expressions, and another spatial frequency corresponds to the same faces with ”happy” or ”surprise” emotions. It means we can have a hybrid face that shows different emotions at different distances.

Another thing is related to color. Color offers a very strong grouping sign that can be used to create more convincing illusions.

Until now, hybrid images were obtained by combining two images, but is it possible to combine more than two images and still have a coherent understanding that changes with changing viewing distances?

**Other ideas:**

1. **Optical Flow Estimation:** It is the problem of finding pixel-wise motions between consecutive images. Approaches for optical flow estimation include correlation-based, block-matching, feature tracking, energy-based, and more recently gradient-based. Using this idea can be useful in finding the amount of pixel relocation in each channel in comparison to other channels.
2. **Second Idea:** We can calculate the sum of all the values in each column and have a value for each. Then order all these values and produce a vector. Then we have a vector for every channel. It may be possible to understand the amount of pixel relocation in each channel in comparison to other channels by the correlations between these tree vectors.