Image Processing - Exercise 3

itaiwar

Introduction

The exercise had two goals:

1. Given two images create a hybrid image out of them.
2. Given two images and a mask, blend the images seamlessly in such a way that in the new image, the area where the black part overlaps the first image only it will be prominent and where the white part of the mask overlaps the second image only it will be prominent.

Both results were achieved using manipulation of Gaussian and Laplacian pyramids, concepts taught in class.

Algorithms

As discussed in class, an image Laplacian pyramid enables the reconstruction of the original image by iteratively summing all its layers after enlargement. Both algorithms leverage this concept but in different ways. In image blending, we leverage this concept by creating blended summands to create a seamlessly blended image. This technique helps make the seams unnoticeable, as they get averaged out across multiple layers during the enlargement process. In the context of hybrid image creation, we take advantage of the Laplacian pyramid construction by alternating between different pyramids – taking the first layers from the image to be seen from afar and the rest from the image to be seen up close. This approach allows us to craft an image that appears differently to the viewer based on the viewing distance, incorporating fine details from the first image and coarse details from the second.

Both algorithms use the same initial construction of Gaussian and later Laplacian pyramids, which I will elaborate on. Some notes beforehand:

- My algorithms apply to single color channels. Should be run multiple times for RGB.
- We require all images and masks to have the same dimensions.
- The mask for image blending should be binary, i.e. contain either zeros or ones.
- The arguments for image blending are two image paths, and a path to a mask.
- The arguments for hybrid image creation are two images a parameter k to denote how many Laplacian layers should be taken by the first image before starting to take layers from the second image, an array of layers to be ignored, and a parameter ℓ for adding a global illumination to the image.
- At the end of these two algorithms, we clip the resulting array to values in $[0,255]$ and then convert them to arrays of type uint8 (to get integers).

Common steps (in bold) and implementation details:

1. **Load the images and the mask (if required) as numpy arrays of type float.**

2. **Calculating the Gaussian pyramid for each image and the mask (if required).**

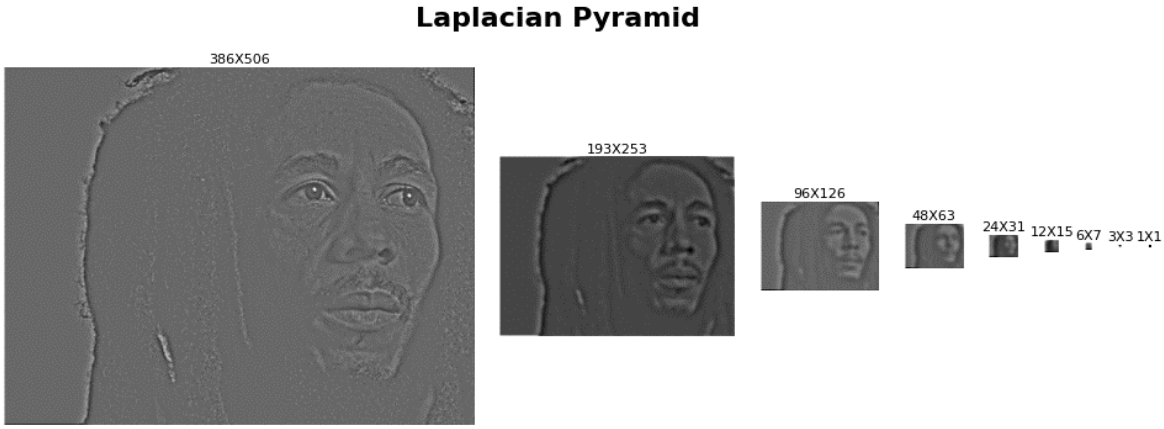
This is achieved by iteratively reducing the topmost layer created thus far by applying a blurring filter on the image (I used a Gaussian kernel with standard deviation of 3) and subsampling every pixel on every second column and second row (starting from the input image itself). If the current layer being reduced is of size 49×41, the reduced layer will be of size 24×20, since integer division by 2 is being performed. Here is an example of a Gaussian pyramid (the first image on the left being the original). We can observe how the more we progress along the pyramid the more blurred the images are, losing detail and averaging out in color.

3. **Determine the dimensions of the different levels of the first image's pyramid and determine whether a column or row of pixels should be added after the expansion.**

As the photos have the same dimensions, the dimensions of the pyramid layers will be the same for all images and the mask, so we arbitrarily choose the first one. This calculation is necessary as the expansion of a layer (in the next step) is not simply doubling its height and width, it needs to consider the pixel which was dropped during the reduction due to the integer division by 2 to restore the dimensions of the layer that came before. To continue the example from the previous step, if my image were of size 49×41, reducing the layer would result in a layer of size 24×20, and a naïve expansion of the reduction would yield a layer of size 48×40 which is different from the original. That is why we must check when a row/column of zeros should be added and take this information into account while expanding.

4. **Calculate the Laplacian pyramids for the two images only.**

This is done by iteratively enlarging layer $i + 1$ of the Gaussian pyramid, subtracting it from Gaussian layer i and storing the result. Here is the corresponding Laplacian pyramid to the Gaussian pyramid I included previously, where we can observe how the more we progress along the pyramid the fine details we began with get less pronounced, or "duller":



From this point on, the algorithms for image blending and hybrid image creation differ.
For image blending:

5. **Create a joint Laplacian pyramid L by combining the Laplacian pyramids of the two images and the Gaussian pyramid of the mask image. Each layer i of the pyramid is computed element-wise as follows:**

$$L[i] \leftarrow (1 - Gaus\_mask[i]) \cdot Laplacian\_im1[i] + Gaus\_mask[i] \cdot Laplacian\_im2[i]$$

This determines where in the resulting blended image the first image will be more prominent and where the second image will be. The initial mask is binary, however each expansion blurs it using a gaussian kernel and the value the pyramid holds in different levels will not be binary. A mask value closer to 1 assigns a higher weight to the Laplacian of the first image, while a value closer to 0 emphasizes the Laplacian of the second image.

6. **Create the blended image by iteratively enlarging a layer of the new Laplacian pyramid and adding it to the layer below it. The result is the blended image.**

For hybrid image creation:

5. **Create the blended image by iteratively enlarging layer i of the first image's Laplacian and adding it to layer $i-1$ for k iterations and doing the rest of the iterations using the Laplacian pyramid of image the second image (this loop should be started from the top layers and progress downwards). The first image is the image which should appear when looked at from afar and the second image is that which should appear when looked at closely.**

I realized that a better result is obtained after allowing the function to skip layers of the Laplacian pyramids (and just enlarge the image to the next level without adding anything to it). In addition, I found that the result may turn out to be darker than anticipated, so I added a parameter of global illumination to the image so the extremely dark patches would become brighter, as well as making the brighter patches less distinct (they max out at 255).
Most of the challenges I faced were technical. For instance, assignment of incorrect types to arrays before I was supposed to: making the array be of type uint8 before clipping to values in $[0,255]$ led to wrapping which caused unexpected dark patches in the output images.

Apart from image loading and basic numpy functions including slicing, I used the gaussian filter and convolve functions provided by scipy.ndimage library. I used numpy for its ease of matrix manipulation and scipy because it provided those two functions.

Results

Hybrid Image (original images were converted to grayscale before running the algorithm):

Result 1 Result 2 Final Result

Result 1: Transition at layer 8 from the top, no illumination added. When shrinking, the fine details are still prominent. The dark color of Bob Marley's hair is too noticeable when viewed from up close.
Result 2: Transition at layer 8 from the top, global illumination of 80 added. The problem with the dark hair was solved, but the fine details are still very prominent.
Final Result: Transition at layer 8 from the top, disregard the 8th layer (leaving only one Laplacian layer from the closer image), global illumination of 80 added.

Image blending:

The binary mask Result with mask Result with inverted mask

Pyramids (I added visualization and explanations in the description of my algorithms)

Conclusion

Creating hybrid images proved to be a more intricate and creative process than I initially thought; it wasn't as straightforward as I expected. It demanded a nuanced understanding of factors that can influence the outcome in each specific instance. I strongly believe that experimenting with various techniques, such as more thoughtful pixel weighting during transitions between layers or image manipulation using a Fourier Transform could potentially yield even more impressive results. Nonetheless, I am pleased with the results I achieved. In contrast to the creation of hybrid images, image blending was rather straightforward. I find satisfaction in the ability produce these results independently.