Image Filtering and Image Pyramids

Alejandro Trujillo
Universidad de los Andes
Bogotá, Colombia
af.trujilloa@uniandes.edu.co

Santiago Martínez
Universidad de los Andes
Bogotá, Colombia
s.martinez1@uniandes.edu.co

Abstract

This report has the intent of familiarizing the reader with image filtering (Low-pass and High-Pass) and image pyramids, more specifically Gaussian and Laplacian pyramids. two similar images were used in order to achieve the expected outcome. These outcomes were a hybrid image and a blended image. To get a hybrid image, a Low-pass filter was applied to one image and a high-pass one to the other. both results were added and normalized. The blended image was made using the different levels of the Laplacian pyramid of halves of both images and adding them up until getting the original resolution.

1. Introduction

In the present laboratory, we want to use Python as a tool to scale, crop and filter images in order to compute hybrid and blended images. The importance of the treatment of these images is to explore the perception of the human vision and, with the scaling, cropping and filtering of this images, create amusing visual illusions. The purpose of overlapping high and low frequencies of different images (hybrid) is to create an optical illusion where an image can be seen when the receptor is up close and another when it's far away. Another task we decide to take, was to create a blended image using image pyramids, which are computed by simply re-scaling and filtering the image multiple times over different levels. For this task, two stock images were used. Two libraries were crucial when taking on this task. These were "openCV", which has a wide content referring to the computer vision and machine learning, and "numpy", which is very useful when it comes to scientific computation and multidimensional objects.

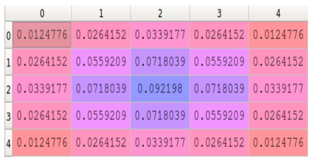
2. Materials and Methods

Two stock images downloaded from "Pexels.com" were used to make the experiments. These images were chosen due to their similarity in visual information and apparent alignment, as these characteristics are needed in order to get a satisfactory result when creating the hybrid image and the blended image. Some processing was made to the image after they were loaded into Python. Firstly, the road picture was cropped in order to get a better alignment (paying special attention to the parallel lines). The image was cropped from column 200 to column 1730 and from row 350 until the end.
After this, both images were resized to 1024x512. This must be done because later we need to compute image pyramids, and this can only be done with images whose dimensions are a power of 2 in order to not complicate the code any further. Moreover, the "road" picture was filtered with a sharpen filter of size 3x3, so the edges are more defined. This is done because the "Bridge" image has much more defined edges, so by doing this the similarity between the two images would increase.

Now, to compute the hybrid image, a low-pass filter must be designed. A Gaussian filter was used for this purpose as it's a well known low-pass filter and it's also widely used. The kernel was designed using the definition of the Gaussian bell in 2 dimensions (equation 1) with a given σ and a given size "k", assuming zero to be in the middle of the window.

$$f(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{\sigma^2}} \tag{1}$$

The kernel was normalized so over-saturation is avoided.



	0	1	2	3	4
0	0.0124776	0.0264152	0.0339177	0.0264152	0.0124776
1	0.0264152	0.0559209	0.0718039	0.0559209	0.0264152
2	0.0339177	0.0718039	0.092198	0.0718039	0.0339177
3	0.0264152	0.0559209	0.0718039	0.0559209	0.0264152
4	0.0124776	0.0264152	0.0339177	0.0264152	0.0124776

Figure 1. Example of a kernel with σ=2 and k=5

A function with all the aforementioned characteristics was made. this function returns a matrix containing the designed Gaussian kernel. The function "filter2D" of the openCV library was used to compute the cross-correlation

between the image and the kernel. The kernel is symmetrical so this calculation is equivalent to the convolution.[3]
for the ”Road” image, a σ of 10 and a size of 11x11 was used for filtering, while for the ”Bridge” image, the values selected were a σ of 8 and a size of 9x9. in the case of the ”Bridge” image, in order to get a high pass filter, the filtered image was subtracted from the original image. After this, the two resulting images were added, normalized and multiplied by 255, so over-saturation is avoided. The result is the hybrid image we were looking for.
In order to compute the blended image, it was necessary to compute the Gaussian and Laplacian pyramids. Two functions of the openCV library were used, pyrUp and pyrDown. In this experiment, the pyramid consisted of 8 levels. This was made by simply iterating starting on the first image and using the following step on the gaussian pyramid (pyrDown) until the end of the pyramid. Now, in order to get the Laplacian pyramid, we start with the last step of the Gaussian pyramid and we used the function pyrUp and subtract the result with the previous step of the Gaussian pyramid. the pyramids were saved in two different list including all 6 levels. after obtaining the Laplacian pyramid for both images, in each level of the this pyramid the two images were cropped to half of their width, one from left to right and the other from right to left. Right after, the two parts were pasted against each other and saved in a new list. In order to obtain the blended image, it was necessary to use the last list computed (the one with the cropped and pasted images in each level of the Laplacian pyramid) and iterate over it from second to last element. Using this last list, the original image is reconstructed, using the function pyrUp to go up in the pyramid. The following python code illustrates how this was done:

```python
blended=blendedList[0]
for idx in range(1,pyramidLevel):
    blended=cv2.pyrUp(blended)
    blended=cv2.add(blended,blendedList[idx])
```

being *”blendedlist”* the one created from the cropped images in each level of the Laplacian pyramid, *”blended”* the resulting blended image and *”pyramidLevel”* the amount of images the pyramid has (declared before as 6). the coding for these steps was made based on the code found in the openCV tutorials webpage.[4]

3. Results

Both results after filtering can be seen in figures 6 and 7. The result for the hybrid image can be seen in figure 8 We can clearly see the high frequencies of the Bridge and the low frequencies of the road overlapped, which is the expected result. From far away the viewer will see the road more clearly (as the low frequencies from that picture are dominant in this case) and from up close, the bridge will be more visible. The blended image using the pyramids for reconstruction can be seen in figure 10, it can be compared to just cropping and pasting the images (figure 9), it can be clearly seen that the border between the two pictures is considerably more blurred in the picture made using the pyramidal reconstruction, which is the expected outcome, as the high frequency information blends together in the pyramidal method. The Laplacian Pyramid for the cropped and pasted images (From which the blended image is computed), can be found from figures 11 to 16, where it can be seen how the resolution is always half from it's predecessor and higher frequencies are found in higher levels.

4. Conclusions

While experimenting with the uses of filtering and image pyramids, the theory behind both can become clearer for someone who is beginning with image processing and computer vision. Moreover, the theory behind image Pyramids is analog to that of neural networks, which are a state of the art tool when it comes to machine learning and, therefore, to computer vision. We can conclude that understanding the theory behind these tools can ease the learn of more complicated methods that use a similar logic.

4.1. References

[1]*”Alley Photography of Brown Wooden Bridge · Free Stock Photo”, Pexels.com, 2019. [Online]. Available: https://www.pexels.com/photo/bridge-path-straight-wooden-2257/. [Accessed: 19- Feb- 2019].*
[2]*”Road Between Pine Trees · Free Stock Photo”, Pexels.com, 2019. [Online]. Available: https://www.pexels.com/photo/road-landscape-nature-forest-39811/. [Accessed: 19- Feb- 2019].*
[3]*”OpenCV: Smoothing Images”, Docs.opencv.org, 2019. [Online]. Available: https://docs.opencv.org/3.1.0/d4/d13/tutorial_py_ filtering.html. [Accessed: 19- Feb- 2019].*
[4]*”Image Pyramids — OpenCV-Python Tutorials 1 documentation”, Opencv-python-tutroals.readthedocs.io, 2019. [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/ py_pyramids/py_pyramids.html. [Accessed: 21- Feb-2019].*

5. Images

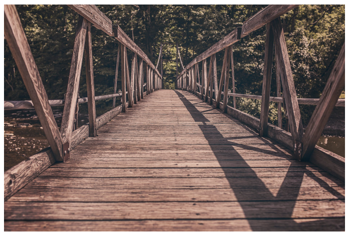


Figure 2. Stock image of a Bridge (1880x1254) [1]

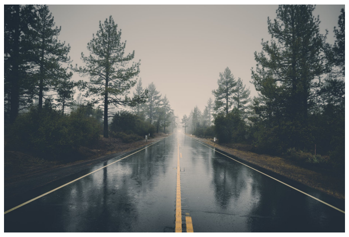


Figure 3. Stock image of a Road (1880x1254) [2]

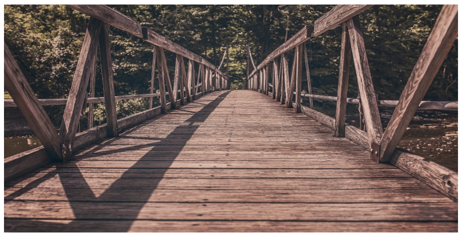


Figure 4. Bridge after resizing

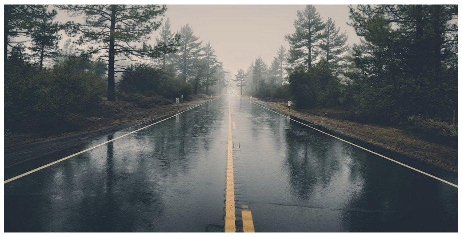


Figure 5. Road after cropping, resizing and sharpening

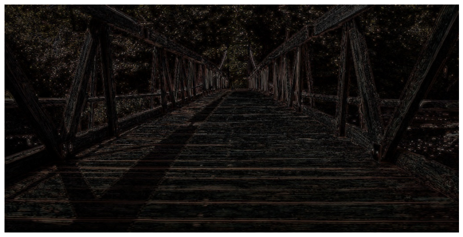


Figure 6. Image with a High-Pass filter (Bridge)

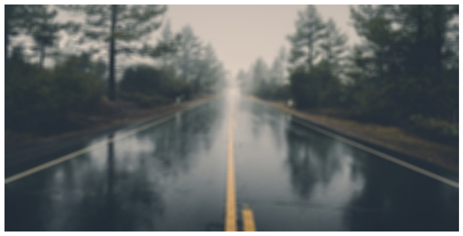


Figure 7. Image with a Low-Pass filter (Road)

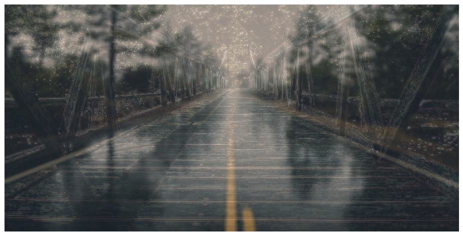


Figure 8. Hybrid Image

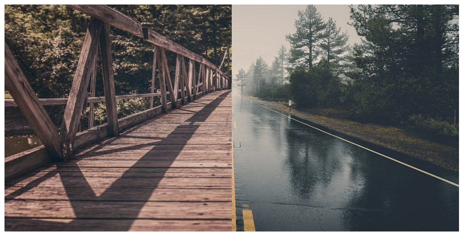


Figure 9. Blended images without using pyramids

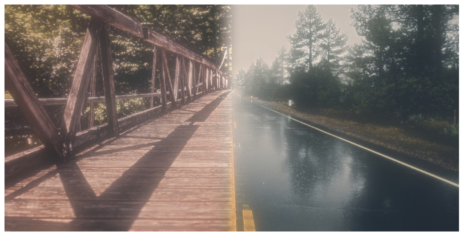


Figure 10. Blended images using pyramids

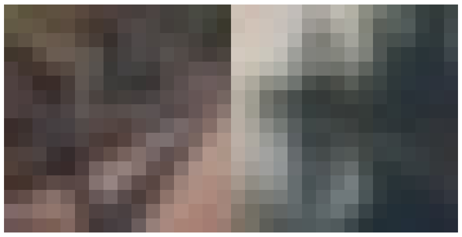


Figure 11. Sixth level of the Laplacian Pyramid of the cropped and pasted images (32x16)

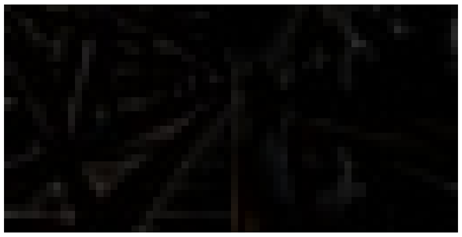


Figure 12. Fifth level of the Laplacian Pyramid of the cropped and pasted images (64x32)

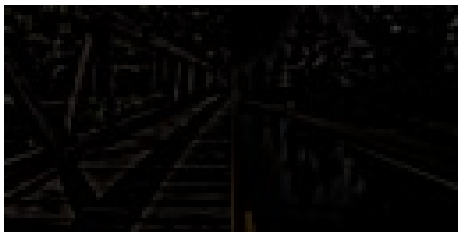


Figure 13. Fourth level of the Laplacian Pyramid of the cropped and pasted images (128x64)

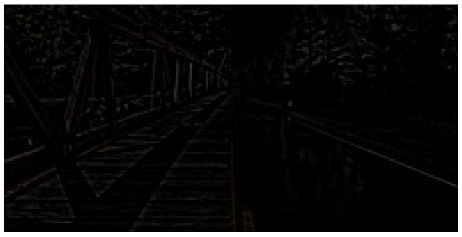


Figure 14. Third level of the Laplacian Pyramid of the cropped and pasted images (256x128)

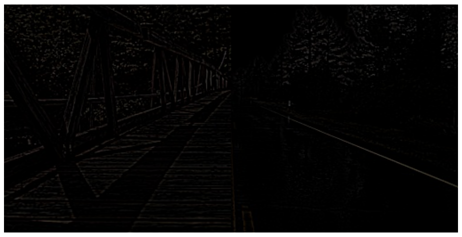


Figure 15. Second level of the Laplacian Pyramid of the cropped and pasted images (512x256)

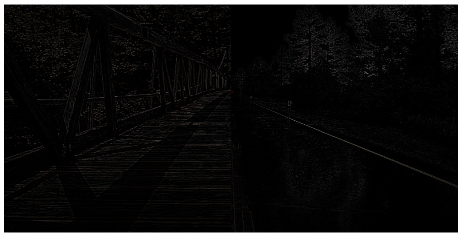


Figure 16. First level of the Laplacian Pyramid of the cropped and pasted images (1024x512)