Laplacian Pyramids and Image Blending
Face Swapping

Mayank Agrawal and Owen Kephart

Who did what for this project?

Mayank implemented the functions necessary for building and reconstructing the Laplacian pyramids, while Owen created the hybrid image and integrated the ellipse widget from cvk2.py into the image blending process. Then, we both worked on creating the alignment program as well as finalizing the parameters used for the hybrid image.

How did you obtain and align your images for each of the two tasks? Did you use any third party software (e.g. Paintbrush, Photoshop), or write a program to help prepare the images or mask?

We created a program called squareImg.py, which enables the user to define a rectangular region in a photo that will be cropped and warped into a 256x256 image. Then, we feed the two identically-sized pictures into pyramids.py. In order to align the faces, we allow the user to draw an ellipse on each face where the blending will occur. The ellipses can have different shapes and origins, because one of the images will go through an affine transform to make sure that the region contained within each ellipse is the same. This functionality allows you to blend faces even if they are different sizes and occur in different locations of the image.

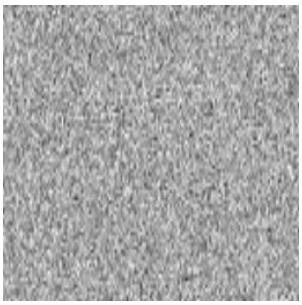
What depth did you choose to build your Laplacian pyramid to, and why?

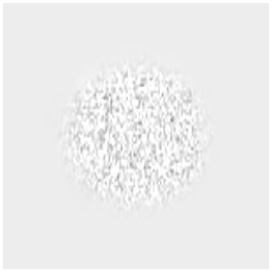
We went to a depth of 8 for our Laplacian pyramids because our picture sizes are 256x256 (2^8 = 256). This way, our smallest pyramid image is a pixel wide, so any larger depth will make no difference. By having the largest effective depth we can, we are able to maximize our spreading of low-frequency content. For example, since the skin color is low-frequency, our images are able to blend the skin tone among two images.

Why does Laplacian pyramid blending blend low-frequency content over a larger distance than high-frequency content? See if you can illustrate this with some carefully chosen input image examples.

In the higher levels of the Laplacian pyramid, only the very low frequency features will still be visible, as there have been several iterations of shrinking and blurring the image. When you blend together two pyramids, the reconstruction will allow low frequency features to spread out farther, as the top level of the blended pyramid (containing only the low frequency features of each) gets upscaled more times than the other levels.

Blending a circular section of high-frequency content (left) onto white yields (right):





On the other hand, blending a circular section of low-frequency content (left) onto white yields (right):

The circle used to blend the images are identical. However, the noisy, high-frequency image's blending is more constrained to the circle which the pure black image's blending is very spread out.

How did you arrive at good values for the constants σA, σB, kA, and kB for the hybrid image generation? Describe the process.

We started by guessing a start value for each of these constants, with a low sigma for each (about 5), and each of the k values set to 1. With this configuration, we found that the output image was too bright, and so we set both of the k values down to .5 to darken up the image a bit. In addition, we were getting much lower frequency features of Taylor Swift's face than we wanted, so she overpowered Kanye's face even at long distance. So we decreased kB down to 1 to only select the very high frequency features (her hair, eyes, and lips), and increased kA to 20 to only select the lowest frequency features of Kanye's face (just the general shape of his face, along with his facial hair). Finally, we made Taylor's features stand out better by increasing kB to 1.5. We played around with other values, but this set seemed to work the best.

If you display your hybrid image at full size on your computer screen, how close do you need to be in order to primarily see image B? How far away do you need to get before you

only see features from image A? Are these distances fairly consistent between you, your lab partner, and any unsuspecting friends you show your image to?

With the image at its true size (256x256), both of us saw Kanye take over at about 4 feet away from the screen. At about 6-7 feet, Taylor's features become very difficult to see for each of us. However, when Owen took his glasses off, he could hardly see Taylor's features at 2 feet away from screen.

What does the Laplacian pyramid of your hybrid image look like?

Shown here are the first 5 levels of the Laplacian pyramid of the hybrid (lower levels become too small to see very well). As you can see, the lowest level only contains the very high frequency features of the hybrid image, and as the depth increases, the higher frequency features fade out, and the low frequency features start to pop out.

In addition to submitting your program and the Laplacian-pyramid-blended output, you should also submit the "traditional" result of directly alpha-blending the two input images, without the pyramid. Hopefully, the pyramid result looks much more convincing than the traditional result.

Below, we blended Kevin Malone from "The Office" with Ted Cruz. The result with the pyramids is on the left, while the traditional result is on the right. We can see that the blending is much better on the left because it spreads out the low-frequency content of skin color.

Examples:



$ python squareImg.py kwOrig.jpg kw.jpg
$ python squareImg.py tsOrig.jpg ts.jpg
$ python pyramids.py ts.jpg kw.jpg



$ python squareImg.py bushOrig.jpg gb.jpg
$ python squareImg.py obamaOrig.jpg bo.jpg
$ python pyramids.py gb.jpg bo.jpg


$ python squareImg.py kwOrig.jpg kw.jpg
$ python squareImg.py tsOrig.jpg ts.jpg
$ python pyramids.py kw.jpg ts.jpg

$ python pyramids.py jb.jpg mc.jpg
(Justin Bieber and Miley Cyrus)