CENG 421 Assignment 2

Cameron Long v00748439

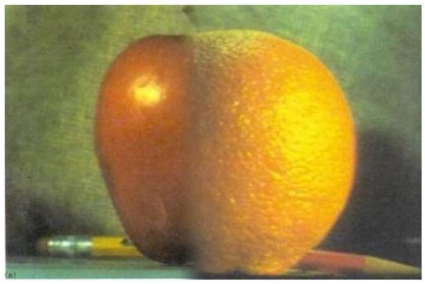
Rationale

I chose the photos I did, simply because I figured you might be able to achieve the same thing as the apple and orange, a reasonably smooth blended image. The two photos of myself are placed similarly in the middle of the frame and would be easy to tell where features have been blended. However, this didn't work out as well as I had hoped.

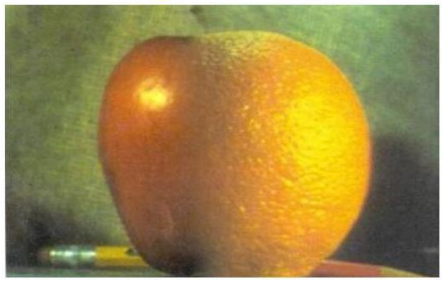
Findings

Listed below are the generated figures from the code. First off is differences in pyramid heights, all with alpha values of 0.4

Fruits:



Height: 4



Height: 6

Height: 8

Self-Portrait:

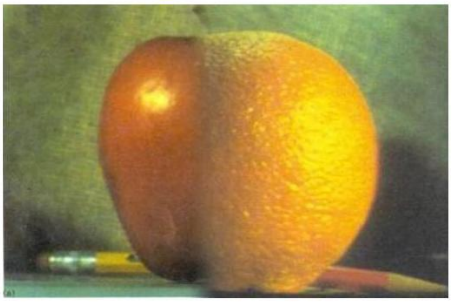
Height: 4

Height: 6

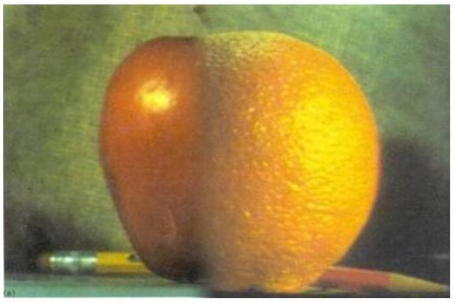
Height: 8

Alpha Values:

Below are the generated images after editing the alpha parameter. The number of levels for this test was 4.



Alpha: 0.4



Alpha: 0.5

Self-Portrait:

These were done with a levels value of 4.

Alpha: 0.4

Alpha: 0.5

Questions:

1.Explain the difference in results when using different values of parameter a for the kernel w,vand when working with pyramids of different heights (number of levels)

To me, the results feels very subtle surrounding the edges of the two images, in theory, the heights value should give us more subsampled images to work with, ie more smaller images to fill in informational blanks with, as we reconstruct the images from its smaller components. In the apples and oranges case, it feels like the parameter levels set to 6 gives the best results, as the apple and orange have a much smoother transition than levels set to 4. Past a certain point, there is no use setting the levels to a higher number, as there will be no image data left for the algorithm to process. In my case, setting the levels past about 10 will generate a null image. (could be an algorithmic error however). The bottom of the image is also not present in higher levels, possibly due to a lack of image information. Higher resolution tests yielded similar results, so I'm unsure if this was supposed to happen or not.

Gaussian filtering works by using the 2D distribution as a point-spread function, so we can normally limit the kernel size to three standard deviations from the mean. In our case, the kernel coefficient diminishes with increasing distance from the kernel's center. Ergo, a higher coefficient will give central pixels more weight in the blurring effects. In the apples and oranges case, it appears to me the colors in the transition zone are more heavily weighted towards red. The differences in the self-portrait pictures cannot really be determined, as the colors are too similar to decide any differences.

2. From a high-level perspective, why does the blended pyramid algorithm work?

When we create a Gaussian pyramid, we're creating a representation of the image using a set of frequency-band images. In our case, we're creating 4, 6 and 8 layers of the pyramid. The first image will represent the low frequency band (with the most detail) and it will be sub-sampled until the last layer. To conserve time complexity, we can take these multiple images and convolute them with a single filter. Each progressive image in the pyramid gets progressively blurrier. (applying a low-pass filter and reducing the size of the image by 0.5). Tie final convoluted set of images are our Gaussian Pyramid.

However, the Gauss pyramid isn't particularly useful by itself, and needs to be combined with another pyramid, the Laplacian Pyramid, a pyramid of band-passed filtered images. Each layer of this pyramid is made by subtracting two consecutive layers of the Gaussian Pyramid. To do this, we need to up sample the image again, to the same size of images in the upper layers. The Laplacian Pyramid contains information in the frequency domain, and we can obtain the spatial-frequency domain.

To reconstruct the image, we sum every layer of the Laplacian pyramid together. Or in our case, we reconstruct two different halves of the image after applying a blur effect to the border of the two images.

The algorithm as is it written in the paper is listed below:

- Step 1: decompose into a set of band-pass filtered component images
- Step 2: component mages in each frequency band are assembled into a bandpass mosaic
    - Joined using a weighted average within a transition zone
- The mosaic images are summed to obtain the final image
    - Blur over borders
- Build Laplacian pyramids LA and LB for images A and B
- Build a Gaussian Pyramid GR for the region R
- Combine LS and LA and LB using nodes of GR as weights, from the paper:
    - LSi(I,j) = GRl(I,j)LA1(I,j) + (1 – GRl9i,j))LB1(I,j)
- Obtain the splined image S by expanding and summing the levels of LS
- The spline is matched to the scale of features within the image themselves, when coarse features occur near borders, they are blended gradually over a relatively large distance without blurring or degrading finer image details along the border.

It works by taking a weighted average of the pyramids across the transition zone to obtain a smoothed over image of both pyramids. The pyramids themselves are convenient for computation and provide low-pass filtering. When the splines are generated, they are matched to the scale of the features that are present within the image themselves and handles coarse features by blending gradually over a relatively large distance without blurring or degrading finer image details.

3.Why do we blend pyramids instead of summing up the left half of image 1 with the right half of image 2, then applying a smoothing filter once?

1) Reduces computational complexity.

- Sample reduction reduces computational complexity
- Gaussian Pyramid is convenient for determining which nodes at each pyramid level lie within the mask area of the image R. and it softens the edges of the mask through an effective low-pass filter
- Avoids artifacts and double exposure effects
- Pyramids are a highly efficient filter
- Only seven arithmetic operations per pixel to produce a full-set of low-pass images.