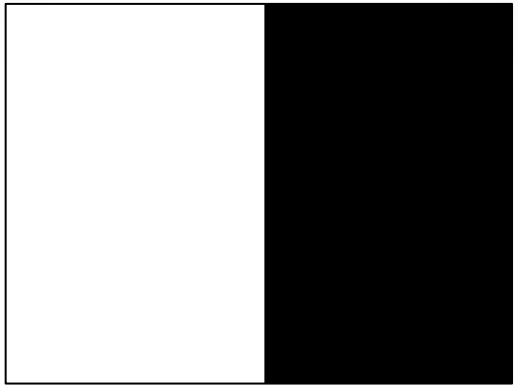
Comp Photography (Fall 2015)
HW 6

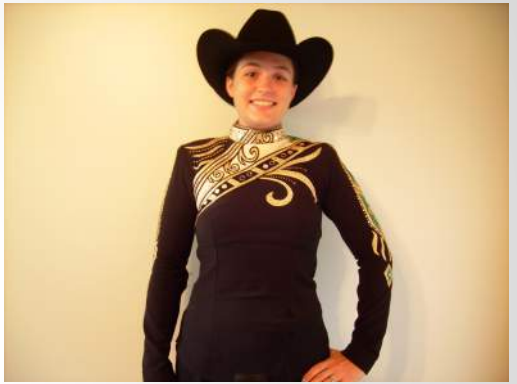
Jacob Kilver

Input

“White” image



Mask image



“Black” image

Function: Reduce

- First, I used two dimensional convolution (scipy's convolve2d function) to convolve the kernel with the image
- Then, I used numpy indexing to specify a step size of two for both the rows and columns of the input image.
- This resulted in an output image with half the number of rows and half the number of columns as the input. This image is a quarter of the input size.

Function: Expand

- This was the opposite of reduce
- I used numpy indexing to copy each point in the original image to every other row and column in the output image.
- The table to the right is representative of the image at this step. The ‘x’ denotes the value from the original image. Zeros fill the remaining positions.

x	0	x	0	x
0	0	0	0	0
x	0	x	0	x
0	0	0	0	0
x	0	x	0	x

Function: Expand

- With the original image mapped to the expanded one, I applied the kernel with 2D convolution
- Finally, since the input image had its information spread out over 4 positions in the output image, I had to multiply the output image by 4 to restore the original information.

Function: gaussPyramid

- To construct the Gaussian pyramid I made repeated calls to the reduce function, appending the output to a python list after each call.
- As such, increasing positions in the list contained smaller images

Function: laplPyramid

- The Laplacian deals with two layers of the Gaussian pyramid at a time.
- To achieve this, I took the first layer of the Guassian pyramid before entering the loop.
- For every layer in the Gaussian pyramid, I expanded the next layer using the expand function

Function: laplPyramid

- Next, I took the difference of the current layer (which was not expanded) and the next layer (which was expanded)
- I appended this difference to a python list

Function: blend

- For each layer of the input pyramids, I took each pixel of the “black” and “white” image and used the mask to determine the resulting layer
- I used the equation given in the comments of the function:

```
output[i, j] = current_mask[i, j]*white_image[i, j] +
               (1 - current_mask[i, j]) * black_image[i, j]
```

Function: blend

- Finally, I appended the output to a numpy list
- This resulted in a new pyramid that had the same dimensions as the input pyramids

Function: collapse

- I began by creating an array the same size as the output image (the array at position zero of the input pyramid). This assured me that I would have the proper sized output.
- Next, I reversed the pyramid since I would need to expand from the smallest array first

Function: collapse

- Then, for each layer of the pyramid, I took the layer and expanded it, making sure to crop the image appropriately
- Finally, I added the expanded image to a running sum that being maintained
- This running sum was returned

Creating a Mask

- I wanted to blend the two different outfits my wife wears for her horse shows into one image
- I wanted to take half from one image and half from the other

Creating a Mask

- To create the mask I read in the “white” image
- I created a mask image the same size as the white image and filled it with zeros
- Next I set one half of the mask image to all have the value 255
- I had to tweak where the division was since the two images were not perfectly centered
- Finally I saved this image for use later

Results

Results