Comp Photography (Fall 2015)
HW 4

Jacob Kilver
GTID: jkilver3

Part 1: Computing Image Derivatives

- Initialized output array to be the same size of the input array minus one in the direction the derivative was being taken.
- For each pixel, calculated the difference between it and the next adjacent pixel in the direction of the derivative
- Inserted difference at appropriate index in output array

Part 2: Computing Gradient

For this part, I took the strategy of ‘sliding’ the kerner over the entire image to compute the gradient.

Part 2: Cont’d

- Initialized output array to the size of the input image minus 2 in both directions (since the kernel size could be assumed to be size 3)
- For every pixel in the input image, I took a ‘cutout’ of the surrounding pixels in the image.

Part 2: Cont’d

- Computed the element-wise product of the kernel with the image cutout.
- Summed all the elements of the matrix and took the absolute value
- Inserted this result into the appropriate index of the output array

Part 2: Results

- The test image included with the assignment was a good image to show the results with
- I used two different kernels: one for the x-direction, one for the y-direction.
- To compute the gradient for the whole image, I summed the results from the individual gradient computations

Part 2: Results - Sobel operator

Image Gradient - x direction

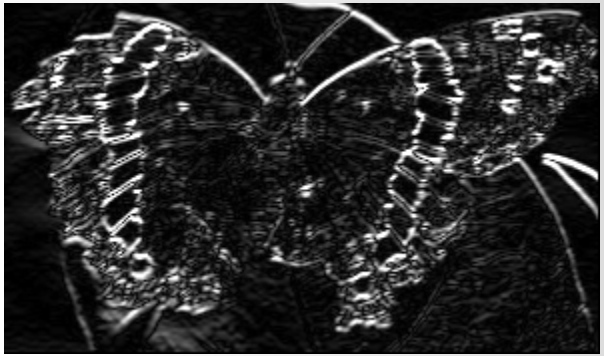

Image Gradient - y direction

Image Gradient - x and y directions

Part 2: Going further

- Use different operators
  - Prewitt
  - Roberts
- Use thresholding
  - This can make lines more definitive and filter out some of the noise by cutting out some edges that are not clearly defined
  - Different values filter out more or less

Part 2: Going further

- Threshold of 128
  - Notice that some of the lines have been removed
  - Others have become brighter