CS 675 – Computer Vision – Spring 2018
Instructor: Marc Pomplun

Assignment #2

Posted on March 1 – Due by March 20, 2pm

Question 1: Implement the Sobel and Canny Edge Detectors

In a first step, write a function that performs a convolution on a given image and filter. Both inputs and the output are of the Matrix type, which means that you first have to convert the input image into a matrix. This seems cumbersome but is more efficient in the long run, because sometimes we want to perform multiple successive convolutions on an image, and we would also like to be able to use real-valued convolution inputs. The function should thus have the following signature:

```
Matrix convolve(Matrix m1, Matrix m2)
```

As you know, `m1` and `m2` are in principle exchangeable, but only if they are padded with zeroes outside of their borders so that all overlaps of elements from `m1` and `m2` are considered. For our computer vision applications, however, we expect the filter to be smaller than the input image, and during convolution we never want any elements of the filter to be outside of the image. Therefore, we require that `m1` contains the values of the input image, and `m2` those of the filter. The resulting matrix is always of the same size as the input image, with zeroes in those positions that could not be reached by the center element of the filter. For example, if the input image has 7 by 7 entries and the filter has 3 by 3 entries, then the output is a 7 by 7 matrix with zeroes in its first and last rows and its first and last columns. The function assumes that we use the center of the filter as the indicator of where to store the output value. If the height or width of the filter is an even number of elements, the center value is rounded down. For example, if `m2` is a 2 by 2 filter, we use its upper-left position as its "anchor" for storing output values.

Having implemented the `convolve` function, it is now easy to write a `sobel` function that takes an input image and outputs a grayscale image of the same size, indicating the edge strength at each position in the input image. The weakest edge pixel in the image has intensity 0 (black), and the strongest one intensity 255 (white), which is accomplished by linearly scaling the Sobel filter edge strength output. Use the following signature:

```
Image sobel(Image img)
```

You are now in a good position to implement the canny function, using the same signature:

```
Image canny(Image img)
```

Include all operations as discussed in class, including initial smoothing and final hysteresis thresholding by determining two thresholds that works for most real-world images. Use the Sobel filter for gradient computation instead of the 2×2 filters we discussed in class. This way, you can mark pixels as edges rather than corners of pixels. The output should be 255 (white) for edge pixels and 0 (black) for non-edges. Finally, write a function `edgeDetection` that reads an image and stores its Sobel and Canny output images onto the hard drive:

```
void edgeDetection(char *inputFilename, char *sobelFilename, char *cannyFilename)
```

You do not have to worry about catching exceptions or similar things. This is not a software engineering class.

Find an image online or from your computer that clearly demonstrates the difference in results between the Sobel and Canny detectors. Put your code, image, and results into your course directory.

Question 2: Find the Circle

Imagine that you found the three edge points (1, 1), (-3, 4), and (-2, -3). Determine the center and radius of the circle that passes through all of these points. Please write down your complete computation.

Question 3: Here Comes the Math

Assume that you found a contour that is precisely shaped like a parabola defined by the equation $y = x^2$. Determine the radius of its osculating circle in the point (1, 1), and the osculating circle in the point (2, 4). Describe how you compute the results.

Question 4: Image Filtering

a) Apply a 3×3 median filter to the 5×5 image below and write down the resulting 5×5 image (with zeroes in those cells that the filter cannot "reach"). Then apply a uniform 3×3 smoothing filter (i.e., one that has the same value in every cell) to the original 5×5 image and again write down the resulting 5×5 image. I recommend that you do this by hand rather than using a computer or calculator.

0	150	0	150	0
0	150	0	150	0
0	150	0	150	0
0	150	0	150	0
0	150	0	150	0

b) If you want to blur an image (i.e., make it look like it were out of focus), which of the two filters would you use? Explain why.

Question 5: Streamlining Sobel

As you know, for the Sobel filter we define edge magnitude as the square root of the sum of the squared outputs of the horizontal and the vertical filters. Now assume that we simplify this definition a bit – edge magnitude is now computed as the sum of the outputs of the two filters.

(a) How could we then make edge detection more efficient by using only one filter instead of two but obtaining exactly the same result for edge magnitude? Determine that filter.

(b) What is the problem with this new definition of edge magnitude?