Homework 1
M1522.001000 Computer Vision (2017 Spring)

2017-81517 Patric Steiner

Date: March 9 Thursday

1 Composing Filters [5 points]

Yes, applying G followed by E is equivalent to applying E and then G, because they are both linear kernels and can be applied by convolution. Convolution is commutative and associative, so the order is irrelevant. If M is used however, the result will not be the same. M is a median filter, so it is non-linear and thus cannot be applied using convolution. The result will vary depeding if E or M is applied first.

2 Decomposing a Steerable Filter [10 pts]

A 2D gaussian filter is symmetrical and thus can be separated into two 1D gaussian (horizontal and vertical) filters:

$$G(x,y) = \frac{1}{2\pi\sigma^2} exp(-\frac{x^2+y^2}{2\sigma^2}) = \frac{1}{2\pi\sigma^2} exp(-\frac{x^2}{2\sigma^2}) * \frac{1}{2\pi\sigma^2} exp(-\frac{y^2}{2\sigma^2}) = G_{\mathrm{x}}(x) * G_{\mathrm{y}}(y) \tag{1}$$

This separation has a great impact on computation speed. Applying a 2D filter as is takes $\mathcal{O}(n^2)$ time, while it takes only $\mathcal{O}(2*n)$ time if the filter is separated.

3 Hough Transform Line Parameterization [5 pts]

The following Matlab script shows how image points (x,y) result in a sinusoid in (roh, theta) Hough space. The first figure shows the effect on the sinusoid of increasing x value, the second shows the effect of increasing y value. We can see that in both cases, the x and y value have direct impact on the amplitude, which is increased as x or y grow. We can see that the period is always the same, it does not change with the image point (x,y).

```
t = [-5:.1:5]; % theta

hold off;
y = 1; % show effects of increasing x value
for x = 1:5:50
    p = x*sin(t)-y*cos(t);
    plot(t, p);
    hold on;
end
```

```
hold off;
x=1; % show effects of increasing y value
for y = 1:5:50
    p = x*sin(t)-y*cos(t);
    plot(t, p);
    hold on;
end
```

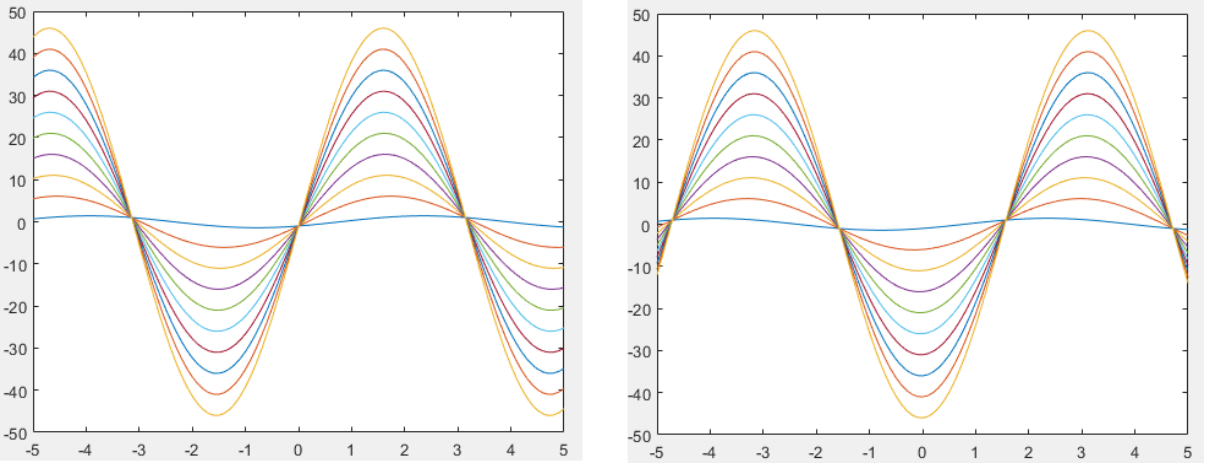

Figure 1: Effect of changing (x,y) values to resulting sinusoid

4 Impulse Response [5 pts]

The Dirac delta function is a function that is 0 everywhere except in a very small part in the middle, there is a spike so big that the area under this spike is exactly 1. The Dirac delta function is the identity value of the convolution operation, which means convolution with the dirac delta functions results in the original function, just like multiplying by 1 or adding 0 for real numbers. It does not change the original.

As a convolution kernel, the dirac delta function is an NxN matrix filled with 0 everywhere except a 1 in the middle. So it is clear that the value in the middle will be weighted 100%, whereas every other value in an iteration has a weight of 0% which means the value just stays the same.

5 Generalized Hough Transform [10 pts]

A parabola can be seen as a "tilted circle". The formula to be used would be $y = \frac{x^2}{4r \sin\theta}$. So we need one more parameter for the angle of the circle.

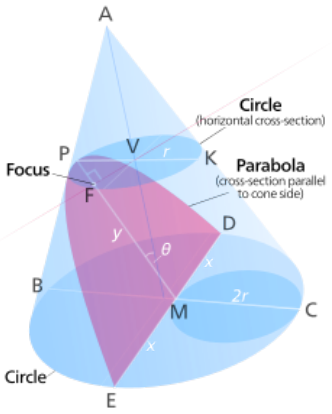

Figure 2: Parabola as a tilted circle in a cone [3]

6 Hough Transform for Line Detection [65 pts]

6.1 Edge Detection [20 pts]

Non maximal supprssion is implemented by iterating through every pixel in the magnitude image and comparing the current pixel to the pixel above and below (left and right respectively). If the current pixel is not the biggest of the three, the pixel is set to 0 to suppress it and make the edge thinner and more accurate.

6.2 The Hough Transform [15 pts]

The parameters used to calcuate these images:

sigma = 7 threshold = 0.3 rhoRes = 1 thetaRes = 1

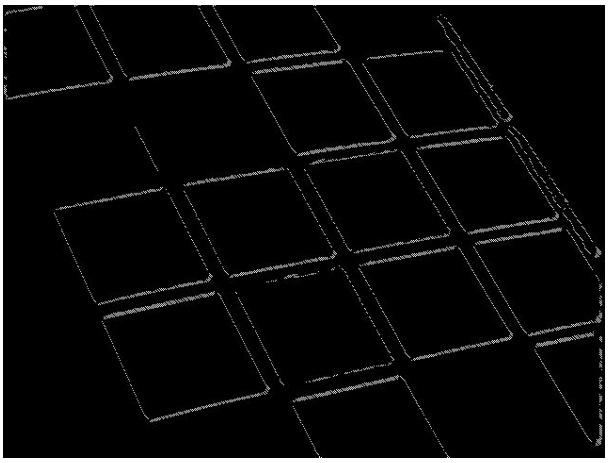

Figure 3: Grayscale image

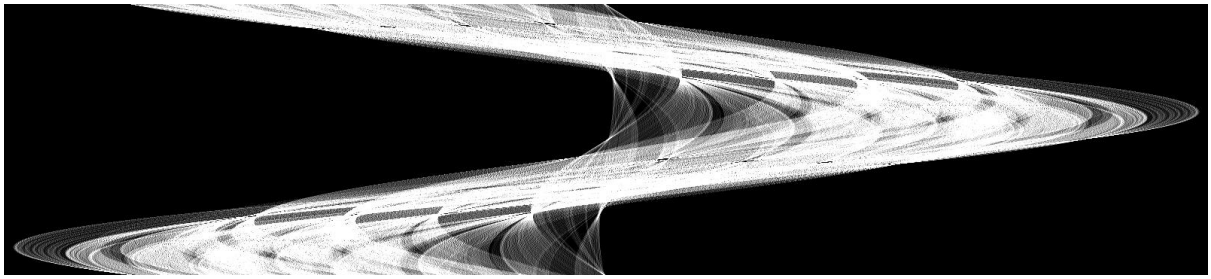

Figure 4: Hough transformation of image (rotated by 90 degree)

6.3 Finding Lines [10 pts]

Non maximal suppression is implemented by iterating through every pixel in the magnitude image and comparing the current pixel to the pixel above and below (left and right respectively). If the current pixel is not the biggest of the three, the pixel is set to 0 to suppress it and make the edge thinner and more accurate.

```
% non maximal suppression
[width, height] = size(Iconv);
for x = 1:width
    for y = 1:height
        p = Im(x, y); % current pixel
        if (x > 1 && x < width) %a void index out of bounds
            if (p < Im(x-1, y) || p < Im(x+1, y))
                Im(x, y) = 0;
            end
        end
        if (y > 1 && y < height) %a void index out of bounds
            if (p < Im(x, y-1) || p < Im(x, y+1))
                Im(x, y) = 0;
            end
        end
    end
end
```

rroh and rtheta are needed to calculate the correct position of the lines.

Unfortunately I did not manage to convert the results of this functions to start and end coordinates of lines, so I was not able to print the lines in the pictures.

References

[1] Canny edge detection. `https://en.wikipedia.org/wiki/Canny_edge_detector`. Accessed: 2017-03-28.

[2] Convolution. https://de.wikipedia.org/wiki/Faltungsmatrix. Accessed: 2017-03-28.

[3] Parabola. https://en.wikipedia.org/wiki/Parabola. Accessed: 2017-03-28.

[4] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.