Homework 2: EECE 5639 - Computer Vision

Sreejith Sreekumar: 001277209

October 2, 2018

1 Question 1

1.1 Code

```
function gray_img = grayscale_generator(w,h)
    gray_img = im2double(128 * ones(w,'uint8'));
end

function sigma = estimate_noise(ImageArray, width, height, count)

    % calculation of E(i,j)
    E = double(zeros(height,width));
    %E = im2double(E)

    for k = 1:count
        for i = 1:width
            for j = 1:height
                E(i,j) = E(i,j) + double(ImageArray(k).Image(i,j));
            end
        end
    end

    E = 1/count * E;

    % calculation of standard deviation of E(i,j)
    sum_sq_diff = 0

    for k = 1:count
        for i = 1:width
            for j = 1:height
                sum_sq_diff = sum_sq_diff + (E(i,j) - double(ImageArray(k).Image(i,j)))^2;
            end
        end
    end

    sigma = sqrt(((1/(count -1)) * sum_sq_diff))
```

```
end

count = 10;
width = 256;
height = 256;


% 10 grayscale images and store in a stucture
ImgArray(1:count) = struct('Image', [], 'Label', '');
for j = 1:count
  ImgArray(j).Image = grayscale_generator(width, height);
  ImgArray(j).Label = j;
end

% add additive zero-mean Gaussian noise to all those images
variance = (2*2)/(255*255)
for j = 1:count
    ImgArray(j).Image = imnoise(ImgArray(j).Image,'gaussian', 0, variance);
end

noise = estimate_noise(ImgArray, width, height, count);
disp("Estimated noise in the images: " + noise)
```

1.2 Results

```
variance =

   6.1515e-05


sum_sq_diff =

     0


sigma =

    2.0071

Estimated noise in the images: 2.0071
```

1.3 Comments

Grayscale images are generated and are stored in a structure arrray. The structure contains the content of the image as a matrix and a label (an integer). Zero mean gaussian noise with a varience of 2.0 is added to every image in the array.

Using the EST_NOISE procedure, the estimated gaussian noise in the image array is around 2.00

2 Question 2: Box Filter

2.1 Code

```
filterSize = 3;
for j = 1:count
    ImgArray(j).Image = imboxfilt(ImgArray(j).Image, filterSize);
end

% use the function implemented for question 1
noise = estimate_noise(ImgArray, width, height, count);
disp("Estimated noise in the images: " + noise)
```

2.2 Results

```
sum_sq_diff =

     0


sigma =

    0.6733

Estimated noise in the images: 0.67326
```

3 Comments

Upon applying the 3 x 3 box filter, the noise reduces from 2.00 to 0.67

4 Question 3

4.1 Code: Generating the 2D gaussian filter mask

```
hsize = 7;
sigma = 1.4;
gaussian_filter = fspecial('gaussian',hsize,sigma)
```

4.2 Result

```
gaussian_filter =

    0.0008    0.0030    0.0065    0.0084    0.0065    0.0030    0.0008
    0.0030    0.0108    0.0232    0.0299    0.0232    0.0108    0.0030
    0.0065    0.0232    0.0498    0.0643    0.0498    0.0232    0.0065
    0.0084    0.0299    0.0643    0.0830    0.0643    0.0299    0.0084
    0.0065    0.0232    0.0498    0.0643    0.0498    0.0232    0.0065
```

0.0030	0.0108	0.0232	0.0299	0.0232	0.0108	0.0030
0.0008	0.0030	0.0065	0.0084	0.0065	0.0030	0.0008

Separating into a horizontal and a vertical filter by the formula:

$$f(x,y) = (\frac{1}{\sqrt{K}}e^{-\frac{x^2}{2\sigma^2}})(\frac{1}{\sqrt{K}}e^{-\frac{y^2}{2\sigma^2}})$$

$$f(x,y) = (\frac{1}{\sqrt{K}}e^{-\frac{x^2}{3.28}})(\frac{1}{\sqrt{K}}e^{-\frac{y^2}{3.28}})$$

$$X = [0.0008\ 0.0030\ 0.0065\ 0.0084\ 0.0065\ 0.0030\ 0.0008]\ and\ Y = X^T$$

Scaling this, we've:

$$X = \frac{1}{36}[1\ 4\ 8\ 10\ 8\ 4\ 1]\ and\ Y = X^T$$

(4) (a) Average Filter

$\frac{1}{5}$

1	1	1	1	1

Image for filtering

		P_1	P_2	P_3	P_4	P_5	P_6		
10	10	10	10	10	40	40	40	40	40

Ignoring the borders and carrying out the computat for the rest of the pixels.

$$P_1 = \frac{50}{5} = 10$$

$$P_2 = \frac{10+10+10+10+40}{5} = \frac{80}{5} = 16$$

$$P_3 = \frac{30+40+40}{5} = \frac{110}{5} = 22$$

$$P_4 = \frac{10+10+40+40+40}{5} = \frac{140}{5} = 28$$

$$P_5 = \frac{10+40+40+40+40}{5} = \frac{170}{5} = 34$$

$$P_6 = \frac{40+40+40+40+40}{5} = \frac{200}{5} = 40$$

10	16	22	28	34	40

$\frac{1}{10}$

1	2	4	2	1

Averaging filter ②

		P_1	P_2	P_3	P_4	P_5	P_6		
10	10	10	10	10	40	40	40	40	40

$$P_1 = \frac{10+20+40+20+10}{10} = \frac{100}{10} = 10$$

$$P_2 = \frac{10+20+40+20+40}{10} = \frac{130}{10} = 13$$

$$P_3 = \frac{10+20+40+80+40}{10} = \frac{190}{10} = 19$$

$$P_4 = \frac{10+20+160+80+40}{10} = \frac{310}{10} = 31$$

$$P_5 = \frac{10+80+160+80+40}{10} = \frac{370}{10} = 37$$

$$P_6 = \frac{40+80+160+80+40}{10} = \frac{400}{10} = 40$$

10	13	19	31	37	40

ⓑ The second filter is much more expensive in computational cost compared the first filter, since it involves more arithmetic operations.

Varience Comparison

Filter 1:

$$\text{Varience} = \frac{\sigma^2}{mn} = \frac{\sigma^2}{5\times 1} = \frac{\sigma^2}{5} \quad \text{—①}$$

Filter 2:

$$\frac{1}{10}[1 \quad 2 \quad 4 \quad 2 \quad 1]$$

Let X_1, X_2, X_3, X_4, X_5 be 5 pixels which hold noise values n_1, n_2, n_3, n_4, n_5

$$O = \frac{1}{10}[X_1+n_1 + 2(X_2+n_2) + 4(X_3+n_3) + 2(X_4+n_4) + (X_5+n_5$$

$$E[O] = \frac{1}{10}[X_1 + 2X_2 + 4X_3 + 2X_4 + X_5]$$

$$D[O] = E[(O - E[O])^2]$$

$$= E\left[\left[\frac{1}{10}(n_1 + 2n_2 + 4n_3 + 2n_4 + n_5)\right]^2\right]$$

$$= \frac{1}{100}\left[En_1^2 + 4En_2^2 + 16En_3^2 + 4En_4^2 + En_5^2\right]$$

Now $E(X^2) = [E(X)]^2 + D(X)$,

$$E(n_1^2) = (E(n_1))^2 + D(n_1) = \sigma^2$$

$$4E(n_2^2) = (2E(n_2))^2 + D(n_2) = 4\sigma^2$$

$E[n_3^2] = E(n_3)^2 + D(n_3) = 16\sigma^2$

$E(n_4^2) = E(n_4)^2 + D(n_4) = 4\sigma^2$

$E(n_5^2) = E(n_5^2) + D(n_5) = \sigma^2$

Now, $\text{Varience} = \frac{(1+4+16+4+1)}{100}\sigma^2$

$= \frac{13\sigma^2}{50}$ —— ②

Comparing the values ① and ②, we can see that the second filter $\left(\frac{1}{10}[1 \; 2 \; 4 \; 2 \; 1]\right)$ causes more varience compared to the first $\left(\frac{1}{5}[1 \; 1 \; 1 \; 1 \; 1]\right)$

(5) Consider that the pixel under consideration is to the right of the strip. We could have four possible cases

	X_1	X_2
50	0	0
50	0	100
50	100	0
50	100	100

↑
Pixel under consideration,

Operator:

-1	2	-1

Possible pixel values after the application of this filter

Case I: $-50+0+0 = -50$

Case II: $-50+0-100 = -150$

Case III: $-50+200+0 = 150$

Case IV: $-50+200-100 = 50$

Now, $P(\text{Salt}) = 0.7$ and $P(\text{Pepper}) = 0.3$

$P(\text{pixel value} = -50) = .3 \times .3 = 0.09$

$P(\text{Pixel value} = -150) = 0.3 \times 0.7 = 0.21$

$P(\text{Pixel value} = 150) = 0.7 \times 0.3 = 0.21$

$P(\text{Pixel value} = 50) = 0.7 \times 0.7 = 0.49$

(6) Equation of the image $I(i,j) = |i-j|$

0	1	2	3	4	5	6	7
1	0	1	2	3	4	5	6
2	1	0	1	2	3	4	5
3	2	1	0	1	2	3	4
4	3	2	1	0	1	2	3
5	4	3	2	1	0	1.	2
6	5	4	3	2	1	0	1
7	6	5	4	3	2	1	0

Applying the 3X3 median filter we have :-
(and copying the borders)

0	1	2	3	4	5	6	7
1	1	1	2	3	4	5	6
2	1	1	1	2	3	4	5
3	2	1	1	1	2	3	4
4	3	2	1	1	1	2	3
5	4	3	2	1	1	1	2
6	5	4	3	2	1	1	1
7	6	5	4	3	2	1	0

(7) 1D step profile $f(i) = \begin{cases} 4 & i \in [0,3] \\ 8 & i \in [4,7] \end{cases}$

4	4	4	4	8	8	8	8

(i) Median filter with $n=3$, and copyiy the border:

4	4	4	8	8	8

(ii) Averaging mask $\frac{1}{4}[1\ 2\ 1]$

4	4	4	4	8	8	8	8
	P_1	P_2	P_3	P_4	P_5	P_6	

$P_1 = \frac{2\times4+1\times4}{4} = 3$ $\qquad P_2 = \frac{4+8+4}{4} = 4$

$P_3 = \frac{4+8+4}{4} = 4$ $\qquad P_4 = \frac{4+16+8}{4} = 7$

$P_5 = \frac{8+16+8}{4} = 8$ $\qquad P_6 = \frac{8+16+8}{4} = 8$

3	4	4	7	8	8

The median filter does not change the pixels in this case whereas the averaging mask smoothens it.