1. Gaussian Filtering (25pt)

|  |  |
| --- | --- |
| **Homework #2**  *Digital Image Processing(EEE5320), 2019-2* | Due Date: 2019. 10. 21 |



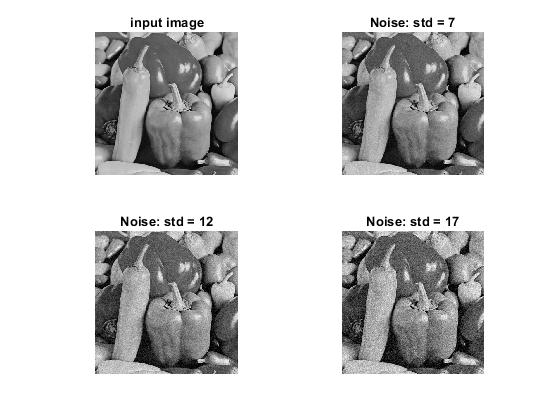
**Figure 1. input1.jpg**

Read the attached “input1.png” and please answer following questions.

* 1. Implement a Matlab function for Gaussian noise and Display the noise images with varying standard deviation (7, 12, 17)

**(You should not use ‘imnoise’ built-in function)**

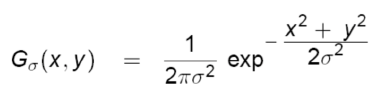
|  |
| --- |
| function output = GaussianNoise (I,SIGMA) %  % random number from SIGMA std Gaussian distrubution  output = I + SIGMA.\*randn(size(I));  end  ***% Complete the remaining part*** |
| **[GaussianNoise.m]** |



* 1. Implement a Matlab function for Gaussian filtering.

**(You should not use ‘imfilter’ built-in function)**

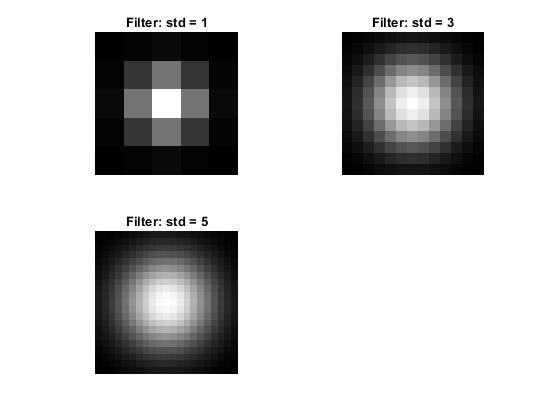
The Gaussian function is as follow:



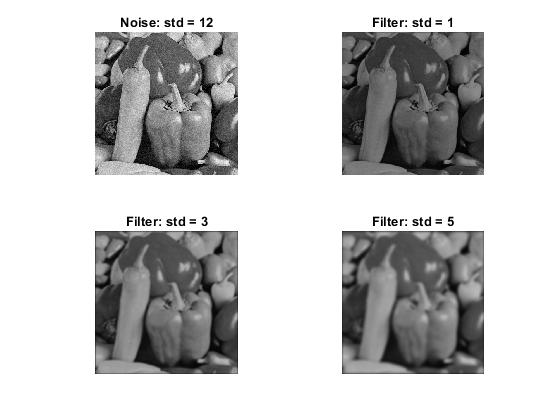
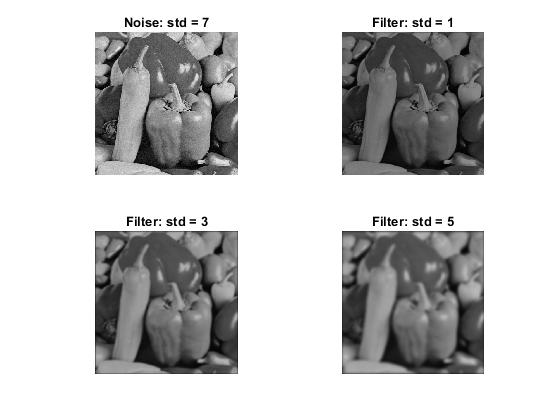
Where  are the location of the pixel and  is a standard deviation

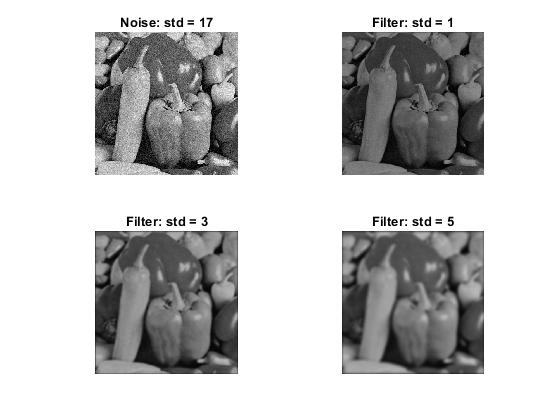
|  |
| --- |
| function output = GaussianFilter (SIGMA) %  % where I is and Gaussian noise images(std : 7,12,17),, SIGMA is standard deviation of Gaussian function  % Each dimension of filter is determined automatically by 2\*ceil(2\*sigma)+1  % filter dimension  G\_len = 2\*ceil(2\*SIGMA)+1;  % make zero matrix corresponding to filter size  G\_mat = zeros(G\_len,G\_len);  % x,y range  range = linspace(-G\_len/2,G\_len/2,G\_len);  % calculate filter matrix values  for x = 1:G\_len  for y = 1:G\_len  % Gaussian distribution  G\_sigma = 1./(2\*pi\*SIGMA\*SIGMA).\*exp(-1 .\* (range(x)\*range(x) + range(y)\*range(y))./(2\*SIGMA\*SIGMA));  G\_mat(x,y) = G\_sigma;  end  end  % return filter matrix  output = G\_mat;  ***% Complete the remaining part*** |
| **[GaussianFilter.m]** |

* 1. Visualize your gaussian filter with standard deviation (1, 3, 5), respectively.



* 1. Display the filtered images with varying standard deviation (1, 3, 5)   
     (input : the gaussian noise images with varying standard deviation (7, 12, 17))





* 1. Explain your implementation and discuss your results

|  |
| --- |
| % image read  im = im2double(imread('input1.jpg'));  % adjust intensity to max 255  im = im(:,:,1) .\* 255;    % parameters  std\_list=[7, 12, 17];  filter\_list = [1, 3, 5];    % 1. add Gaussian noise  figure(1);  subplot(2,2,1);  imshow(im ./ 255);  title('input image');  im\_Trs = cell(3,1);  % calculate noise  for i = 1:length(std\_list)  im\_Tr = GaussianNoise(im, std\_list(i));  % store image matrix  im\_Trs{i} = im\_Tr;  subplot(2,2,i+1);  % show image correspoiding to std value  imshow(im\_Tr ./ 255);  title(['Noise: std = ',num2str(std\_list(i))]);  end    % 2. make Gaussian filter  figure(2);  g\_filters = cell(3,1);  for i = 1:length(filter\_list)  g\_filter = GaussianFilter(im,filter\_list(i));  g\_filters{i} = g\_filter;  % plot filter matrix  subplot(2,2,i);  % normalize intensity for visualization  imshow(g\_filter,[min(min(g\_filter)),max(max(g\_filter))]);  title(['Filter: std = ',num2str(filter\_list(i))]);  end    % 3. filter noisy images  for i = 1:length(std\_list)  im\_Tr = im\_Trs{i};  % plot noisy image  figure(2+i);  subplot(2,2,1);  imshow(im\_Tr ./ 255);  title(['Noise: std = ',num2str(std\_list(i))]);  for j = 1:length(filter\_list)  g\_filter = g\_filters{j};  % apply filter using convolution function  % padding = 'same' for set output size = input size  w = conv2(im\_Tr,g\_filter,'same');    % plot filtered image  subplot(2,2,j+1);  imshow(w ./ 255);  title(['Filter: std = ',num2str(filter\_list(j))]);  end  end |
| **[hw2\_1.m]** |

1. Bilateral Filtering (25pt)



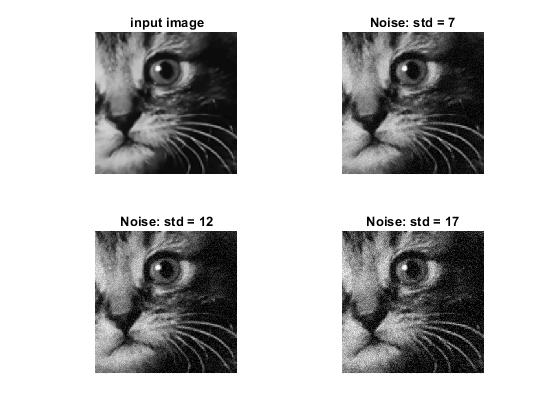
**Figure2. cat\_crop.png**

Read the attached “cat\_crop.png” and please answer following questions.

* 1. Implement a Matlab function for Gaussian noise and Display the noise images with varying standard deviation (7, 12, 17)

**(You should not use ‘imnoise’ built-in function)**

|  |
| --- |
| function output = GaussianNoise (I,SIGMA) %  % random number from SIGMA std Gaussian distrubution  output = I + SIGMA.\*randn(size(I));  end  ***% Complete the remaining part*** |
| **[GaussianNoise.m]** |



* 1. Implement a Matlab function for Bilateral filtering.

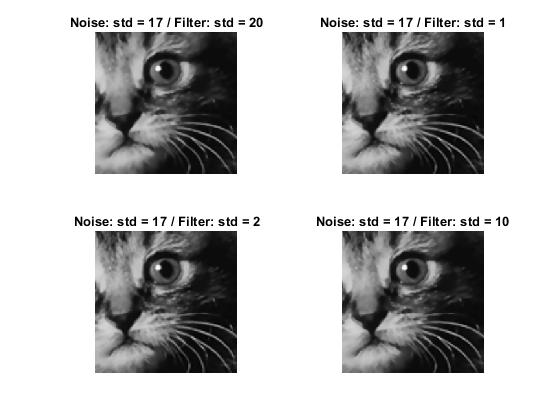
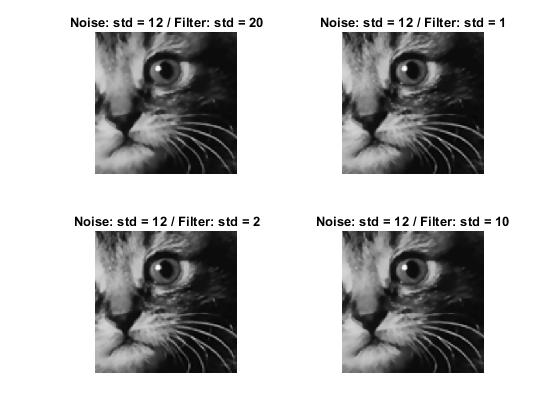
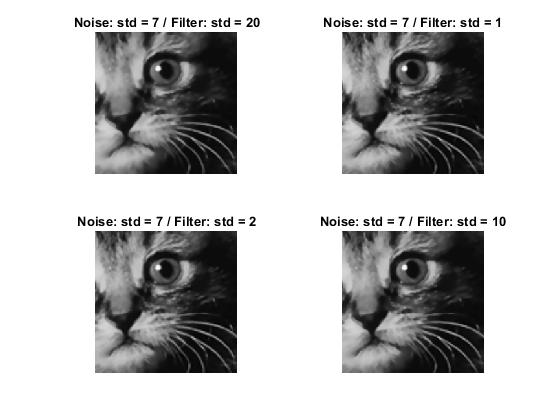
**(You should not use ‘imfilter’ built-in function)**

The output of bilateral filtering is as follow:

Where are neighboring pixels centered on in the kernel and is the intensity value of a pixel.  is a standard deviation. The normalization term .

|  |
| --- |
| function output = BilateralFilter (I,SIGMA\_d, SIGMA\_r, kernel\_size) %  % where I is and Gaussian noise images(std : 7,12,17), SIGMA\_d and SIGMA\_r are standard deviations of Bilateral function. Kernel size n is nxn box.  %padding  pad\_len = floor(kernel\_size/2);  I\_pad = padarray(I, [pad\_len pad\_len]);  % size of padded image  shape\_pad = size(I\_pad);  % calculate Gaussian bilateral filter function  I\_b = zeros(size(I));  % padded image  for r\_I = 1+pad\_len : shape\_pad(1)-pad\_len  for c\_I = 1+pad\_len : shape\_pad(2)-pad\_len  W\_p = 0;  I\_p = 0;  % kernel  for r\_k = r\_I-pad\_len:r\_I+pad\_len  for c\_k = c\_I-pad\_len:c\_I+pad\_len  % calculate w value  w = exp(-1\*((r\_I-r\_k)^2+(c\_I-c\_k)^2) / (2\*(SIGMA\_d^2)) - ((I\_pad(r\_I,c\_I)-I\_pad(r\_k,c\_k))^2 / (2\*(SIGMA\_r^2))));  % sum w values in the kernel  W\_p = W\_p + w;  I\_p = I\_p + I\_pad(r\_k,c\_k)\*w;  end  end  I\_b(r\_I-pad\_len,c\_I-pad\_len) = I\_p / W\_p;  end  end  % return  output = I\_b;  end  ***% Complete the remaining part*** |
| **[BilateralFilter.m]** |

* 1. display the filtered images with varying standard deviation = 20 and = (1, 2, 10) when all kernel sizes are 5.



* 1. Explain your implementation and discuss your results

|  |
| --- |
| % image read  im = im2double(imread('cat\_crop.png'));  % adjust intensity to max 255  im = im(:,:,1) .\* 255;    % parameters  std\_list=[7, 12, 17];  filter\_list = [20, 1, 2, 10];    % 1. add Gaussian noise  figure(1);  subplot(2,2,1);  imshow(im ./ 255);  title('input image');  im\_Trs = cell(3,1);  % calculate noise  for i = 1:length(std\_list)  im\_Tr = GaussianNoise(im, std\_list(i));  % store image matrix  im\_Trs{i} = im\_Tr;  subplot(2,2,i+1);  % show image correspoiding to std value  imshow(im\_Tr ./ 255);  title(['Noise: std = ',num2str(std\_list(i))]);  end    % 2. filter noisy images  for i = 1:length(std\_list)  im\_Tr = im\_Trs{i};  % plot noisy image  figure(1+i);  for j = 1:length(filter\_list)  % apply filter  im\_b = BilateralFilter(im,filter\_list(j),filter\_list(j),5);    % plot filtered image  subplot(2,2,j);  imshow(im\_b ./ 255);  title(['Noise: std = ',num2str(std\_list(i)),' / Filter: std = ',num2str(filter\_list(j))]);  end  end |
| **[hw2\_2.m]** |

* 1. Compare the results for the Gaussian filter with the bilateral filter and discuss the results.

1. Median Filtering (25pt)



**Figure 3. input1.jpg**

Load the attached “input1.png” and please answer following questions.

* 1. Implement a Matlab function for Median filtering.

**(You should not use ‘imfilter’ built-in function)**

|  |
| --- |
| function output = MedianFilter (I, kernel\_size) %  % where I is and input image, and kernel\_size n is nxn size of the filter  %padding  pad\_len = floor(kernel\_size/2);  I\_pad = padarray(I, [pad\_len pad\_len]);  % size of padded image  shape\_pad = size(I\_pad);  % make kernel function  for r\_I = 1+pad\_len : shape\_pad(1)-pad\_len  for c\_I = 1+pad\_len : shape\_pad(2)-pad\_len  % get values in the filter  k\_values = I\_pad(r\_I-pad\_len:r\_I+pad\_len,c\_I-pad\_len:c\_I+pad\_len);  % input median value to padded image  I\_pad(r\_I,c\_I) = median(median(k\_values));  end  end  % return  output = I\_pad;  end  ***% Complete the remaining part*** |
| **[MedianFilter.m]** |

* 1. Add salt and pepper noise to the image with the percentage of spikes as from 10% to 50% by step of 10% **(You should not use ‘imnoise’ built-in function)**

|  |
| --- |
| function output = SaltAndPepper(img, ND)  % Add salt and pepper noise to image of certain density  % ND = noise density, if ND is 0.2, noisy image has 20% noise  % make noise vector  noise\_size = size(img);  sp\_vec = ones(noise\_size(1) \* noise\_size(2),1);  % sample vector index according to noise points  sp\_noise\_idx = randsample(length(sp\_vec),round(length(sp\_vec)\*ND));  % separate white and black noise index  sp\_noise0 = randsample(sp\_noise\_idx,round(length(sp\_noise\_idx)\*0.5));  sp\_noise1 = setdiff(sp\_noise\_idx,sp\_noise0);  % input noise value  sp\_vec(sp\_noise0) = 0;  sp\_vec(sp\_noise1) = 255;  % apply noise value to image  sp\_noise = reshape(sp\_vec,noise\_size);  % return  output = img .\* sp\_noise;  end  ***% Complete the remaining part*** |
| **[SaltAndPepper.m]** |

* 1. Apply median filtering function to the corrupted images with varying the size of filter ([3,3], [5,5], [7,7]) and display the corrupted images and the filtered results.
  2. Explain your implementation and discuss your results.

|  |
| --- |
| % image read  im = im2double(imread('input1.jpg'));  % adjust intensity to max 255  im = im(:,:,1) .\* 255;    % parameters  noise\_list=[0.1, 0.2, 0.3, 0.4, 0.5];  filter\_list = [3, 5, 7];    % 1. add Salt&Pepper noise  figure(1);  subplot(2,3,1);  imshow(im ./ 255);  title('input image');  im\_Trs = cell(length(noise\_list),1);  % calculate noise  for i = 1:length(noise\_list)  im\_Tr = SaltAndPepper(im, noise\_list(i));  % store image matrix  im\_Trs{i} = im\_Tr;  subplot(2,3,i+1);  % show image correspoiding to noise value  imshow(im\_Tr ./ 255);  title(['ND : ',num2str(noise\_list(i))]);  end    % 2. filter noisy images  for i = 1:length(noise\_list)  im\_Tr = im\_Trs{i};  % plot noisy image  figure(1+i);  subplot(2,2,1);  imshow(im\_Tr ./ 255);  title(['ND : ',num2str(noise\_list(i))]);    for j = 1:length(filter\_list)  % apply filter  im\_m = MedianFilter(im,filter\_list(j));  % plot filtered image  subplot(2,2,j+1);  imshow(im\_m ./ 255);  title(['ND : ',num2str(noise\_list(i)),' / Filter : [',num2str(filter\_list(j)),',',num2str(filter\_list(j)),']']);  end  end |
| **[hw2\_3.m]** |

1. Unsharp masking (25pt)



**Figure 4. eye.png**

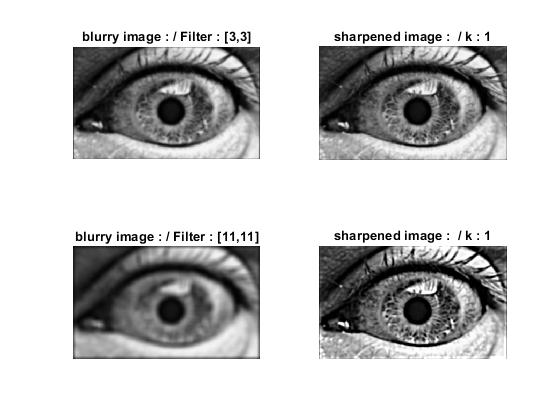
Read the attached “eye.png” and please answer following questions.

* 1. Implement a Matlab function for unsharp mask filter. The output of unsharp mask filtering is as follow

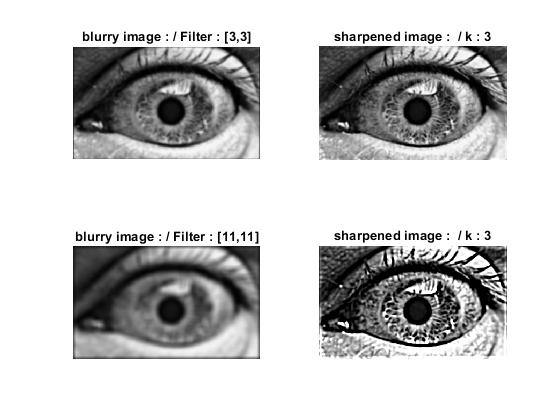
where means blurred image.

|  |
| --- |
| function output = UnsharpMask (I, I\_blur, k) %  % where I is and input image, and I\_blur is blurred image and k is boosting factor  % apply unsharp masking filter  output = I + k .\* (I - I\_blur);  end  ***% Complete the remaining part*** |
| **[UnsharpMask.m]** |

* 1. Display the blurred image and sharpened images using unsharp mask filter with blurred image by average filter as average filter size = [3, 3], [11, 11] and k=1.



1. Set k = 3 and repeat ii.



1. Explain your implementation and discuss your results.

|  |
| --- |
| % image read  im = im2double(imread('eye.png'));  % adjust intensity to max 255  im = im(:,:,1) .\* 255;    % parameters  filter\_list = [3, 11];  k\_list = [1, 3];  % 2. make unsharp masking filter  for i = 1:length(k\_list)  for j = 1:length(filter\_list)  % make blurry image  % make avg filter  avg\_filter = ones([filter\_list(j),filter\_list(j)]);  % apply avg filter to image  im\_blurry = conv2(im,avg\_filter,'same') ./ numel(avg\_filter);    % apply unsharp maks filter  im\_unmask = UnsharpMask(im,im\_blurry,k\_list(i));    % plotting  figure(i);  % plot blurry image  subplot(2,2,2\*j-1);  imshow(im\_blurry ./ 255);  title(['blurry image : ', '/ Filter : [',num2str(filter\_list(j)),',',num2str(filter\_list(j)),']']);    % plot sharpened image  subplot(2,2,2\*j);  imshow(im\_unmask ./ 255);  title(['sharpened image : ', ' / k : ',num2str(k\_list(i))]);  end  end |
| **[hw2\_4.m]** |