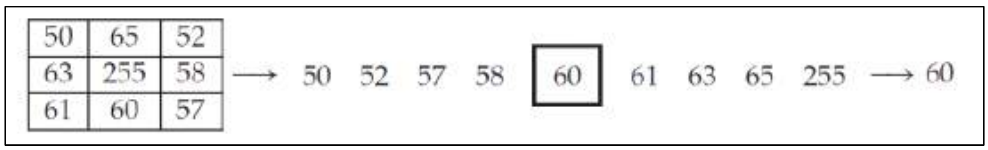
컴퓨터 비젼

[HW07]

학번	201203393
분반	00
이름	김헌겸
과제번호	07

제출일 : 2016년 5월 14일 토요일

1. (MATLAB) Implemenet 3x3 and 5x5 median filter.



median filtering은 window의 값들이 정렬된 상태에서 가장 중앙에 있는 값을 이미지에 적용시키는 filtering이다.

```
function output = get_median_value_from_selection_sort(input)

input_size = size(input);
total_size = input_size(1) * input_size(2);
result = zeros(1, total_size);

for i=1 : total_size
    result(1, i) = input(i);
end

for i=1 : total_size-1
    min_index = i;

    for j=i : total_size
        if result(min_index) > result(j)
           min_index = j;
        end
    end

    temp = result(i);
    result(i) = result(min_index);
    result(min_index) = temp;
end

output = result(1, round(total_size/2));
```

우선 주어진 window에 대해서 값들을 정렬하고 가장 중앙에 위치하는 값을 반환하는 함수를 작성한다. 정렬의 방법에는 Selection Sort를 사용했다. 그리고 반환할 때는 전체 사이즈중 절반에 위치한 데이터를 반환하도록 한다.

```
function out = median_filter(im, filter)

im_size = size(im);
f_h = floor(filter(1)/2);
f_w = floor(filter(2)/2);

out = zeros((im_size(1) + f_h*2), (im_size(2) + f_w*2));
out(1+f_h:im_size(1)+f_h, 1+f_w:im_size(2)+f_w) = im;

temp = zeros(im_size);

for i=1 : im_size(1)
    for j=1 : im_size(2)
        window = out(i : 1 : 2*f_h+i , j : 1 : 2*f_w+j);
        temp(i,j) = get_median_value_from_selection_sort(window);
    end
end

out = temp;
```

이렇게 작성한 함수를 활용하면 median filter를 구현할 수 있다. zero padding이 적용된 이미지에 대해 window마다 median 함수를 호출하여 해당 위치의 픽셀 값을 바꿔버리면 손쉽게 median filter를 구현 할 수 있다.

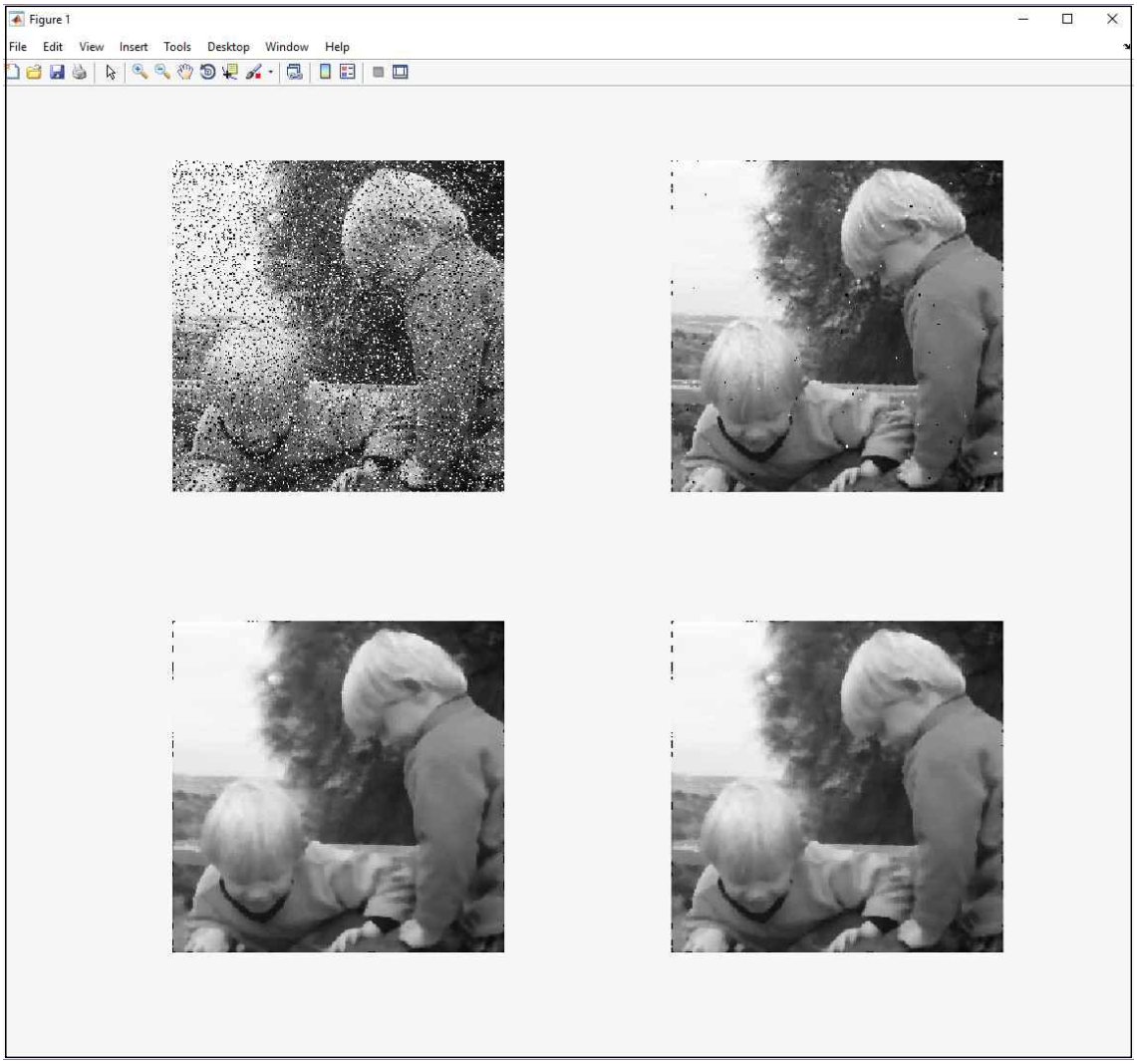
```
function hw_1()

im = imread('twins.tif');
rgb = rgb2gray(im);
noised = imnoise(rgb, 'salt & pepper', 0.2);

filtered_three = median_filter(noised, [3,3]);
filtered_five = median_filter(noised, [5,5]);
built_filtered = medfilt2(noised, [5,5]);

figure,
subplot(2,2,1), title('noised image'), imshow(uint8(noised))
subplot(2,2,2), title('filtered 3x3 image'), imshow(uint8(filtered_three))
subplot(2,2,3), title('filtered 5x5 image'), imshow(uint8(filtered_five))
subplot(2,2,4), title('median built-in'), imshow(uint8(built_filtered))
```

그리고 이렇게 작성한 median filter에 대해서 3x3, 5x5 크기의 window로 연산해보고 그 결과를 분석해본다.



위의 그림에서 순서대로 나타내자면 salt & pepper noise가 적용된 사진이 먼저 나오고 3x3, 5x5, 그리고 matlab에 이미 존재하는 medfilt2 함수를 가지고 비교하는 모습이다. 윈도우의 크기를 키울수록 noise의 제거는 좀 더 잘 되지만 이미지가 약간씩 흐려지고 있음을 알 수 있다.

2. (MATLAB) Implement adaptive filtering.

1. Generate a noisy image by adding the Gaussian noise to 'twin.tif' (grayscale image)
   `imnoise(t, 'gaussian', 0, 0.005)`
2. Obtain $m(x,y)$ and $\sigma_f{}^2(x,y)$ for each pixel (x,y) when 7 × 7 filtering mask is used. The filtering mask is an uniform average filter.
3. Obtain n by taking the mean of all values of $\sigma_f{}^2(x,y)$ over the entire image.
4. Perform the following equation about adaptive filtering

$$m_2(x,y) = m(x,y) + \frac{\max(0, \sigma_f{}^2 - n)}{\max(\sigma_f{}^2, n)}(I_G(x,y) - m(x,y))$$

문제 조건에 명시되어 있는 것을 따라 adaptive filetering을 구현한다.

```
function [aver_result,out] = adaptive_filter(im)

im_size = size(im);

filter = [7,7];

f_h = floor(filter(1)/2);
f_w = floor(filter(2)/2);

out = zeros((im_size(1) + f_h*2), (im_size(2) + f_w*2));
out(1+f_h:im_size(1)+f_h, 1+f_w:im_size(2)+f_w) = im;

temp = zeros(im_size);
dist = zeros(im_size);
aver = zeros(im_size);

for i=1 : im_size(1)
    for j=1 : im_size(2)
        window = out(i : 1 : 2*f_h+i , j : 1 : 2*f_w+j);
        aver(i,j) = mean(mean(window));
        window_quad = window.^2;
        aver_quad = aver(i,j)^2;

        quad_sum = sum(sum(window_quad));
        quad_aver = quad_sum/(filter(1) * filter(2));
        dist(i,j) = quad_aver - aver_quad;
    end
end

n = mean(mean(dist));

for i=1 : im_size(1)
    for j=1 : im_size(2)
        temp(i,j) = aver(i,j) + ((max(0, (dist(i,j)-n)) / max(dist(i,j) , n))) * (im(i,j) - aver(i,j));
    end
end

aver_result = aver;
out = temp;
```

우선 패딩을 적용해준 뒤, mean함수를 이용해서 각 윈도우마다의 pixel마다 평균값을 매트릭스 형태로 만든다. (aver(i,j) = mean(mean(window))) 그리고 윈도우의 제곱에 대한 평균을

구하여 (quad_aver = quad_sum/((filter(1) * filter(2))) 이 둘을 빼는 과정을 통해 variance 매트릭스를 구할 수 있다.

그러고 나서 마지막에 4번의 식을 적용하는 것만으로 adaptive filtering을 쉽게 만들어 볼 수 있다.

```
function hw_2()

im = imread('twins.tif');
gray = rgb2gray(im);
noised = imnoise(gray, 'gaussian', 0, 0.005);

[aver, out] = adaptive_filter(noised);
wiener_result = wiener2(noised, [7,7]);

figure,
subplot(2,2,1), title('average of image'), imshow(uint8(aver))
subplot(2,2,2), title('output of adaptive filter'), imshow(uint8(out))
subplot(2,2,3), title('wiener2 output'), imshow(wiener_result)
```

이번에도 역시 noise가 적용된 이미지에 대해서 직접 구현한 adaptive filter와 built-in인 wiener2 함수를 사용하여 restoration을 적용하고 두 결과물을 출력해 보이면 이와 같다.

되게 흐릿했던 이미지에 대해서 경계 값들이 어느 정도 구분이 되고 꼬마의 여러 부분이 구분이 될 정도로 선명해 져 있음을 알 수 있다.