Project2

Student ID: 311512019   Name: 江柏霖

a. Source codes

```
%% Clear the environment and the command line
clc;
close all;
clear;

%% read image
kid_img = imread('Kid2 degraded.tiff');
[m,n] = size(kid_img);

%% denoise by alpha-trimmed mean filter
size = 5;
alpha = 16;

% expand original image using symmetry method
r = floor(size/2);
kid_img_ex = zeros(m + 2*r, n + 2*r);
kid_img_ex(r+1:end-r ,r+1:end-r) = kid_img(:,:);
for i = 1:2
    kid_img_ex(i,r+1:end-r) = kid_img(r-i+1,:);
    kid_img_ex(r+1:end-r,i) = kid_img(:,r-i+1);
    kid_img_ex(end-r+i,r+1:end-r) = kid_img(end-i+1,:);
    kid_img_ex(r+1:end-r,end-r+i) = kid_img(:,end-i+1);
end

% apply alpha-trimmed mean filter
kid_img_denoise = zeros(m,n);
for i = r+1 : r+m
    for j = r+1 : r+n
        tmp = kid_img_ex(i-r : i+r, j-r : j+r);
        tmp = reshape(tmp,[1,size*size]);
        tmp = sort(tmp);
        tmp = tmp(alpha/2+1:end-alpha/2);
        kid_img_denoise(i-r, j-r) = sum(tmp)/(size^2-alpha);
    end
end

kid_img_denoise = uint8(kid_img_denoise);
```

```
%% inverse filyering
%set Inverse Gaussian LPF
IGLPF = zeros(2*m,2*n);
kid_img2 = zeros(2*m,2*n);
D0 = 250;
for i = 1:2*m
    for j = 1:2*n
        d = (i-m).^2+(j-n).^2;
        IGLPF(i,j) = exp(d/2/D0/D0);
        if(i<=m && j<=n)
            kid_img2(i,j) = kid_img_denoise(i,j);
        end
    end
end

% centering processing, multiply(-1)^(x+y)
for i = 1:2*m
    for j = 1:2*n
        kid_img2(i,j) = kid_img2(i,j)*((-1)^(i+j-2));
    end
end

%set Butterworth LPF
BLPF = zeros(2*m,2*n);
Beta_b = 0.414;
n_b = 10;
D0_b = 300;
for i = 1:2*m
    for j = 1:2*n
        d = (i-m).^2+(j-n).^2;
        BLPF(i,j) = 1/(1+Beta_b*(d/D0_b/D0_b)^n_b);
    end
end

% apply inverse filter
kid_inverse_img = real(ifft2(fft2(kid_img2).*IGLPF.*BLPF));

% crop M*N image
kid_final_img = zeros(m,n);
for i = 1:m
    for j = 1:n
        kid_final_img(i,j) = kid_inverse_img(i,j)*((-1)^(i+j-2));
    end
end

%% output

figure(1);
imshow(kid_img,[]);

figure(2);
imshow(kid_img_denoise,[]);
fig= gcf;
exportgraphics(fig,'kid_img_denoise.png','Resolution',200);

figure(3);
imshow(kid_final_img,[]);
fig= gcf;
exportgraphics(fig,'kid_final.png','Resolution',200);
```

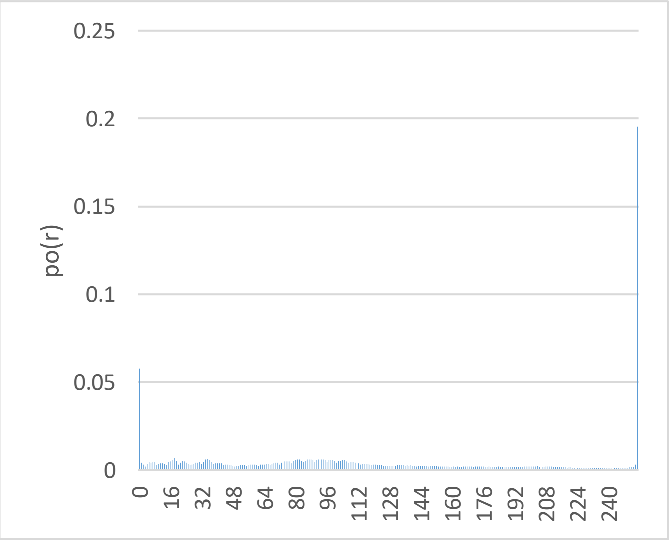
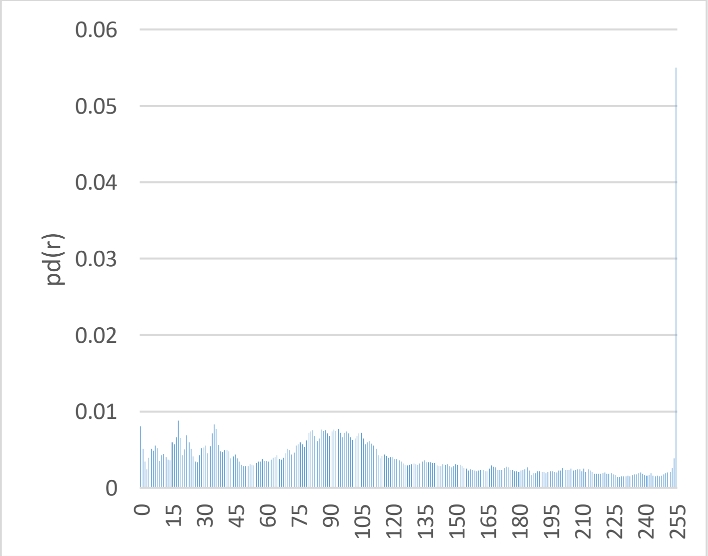
b. Results of noise model and model parameters

The noise model is Salt-and-Paper noise.

By compare the histogram of original image and de-noised image

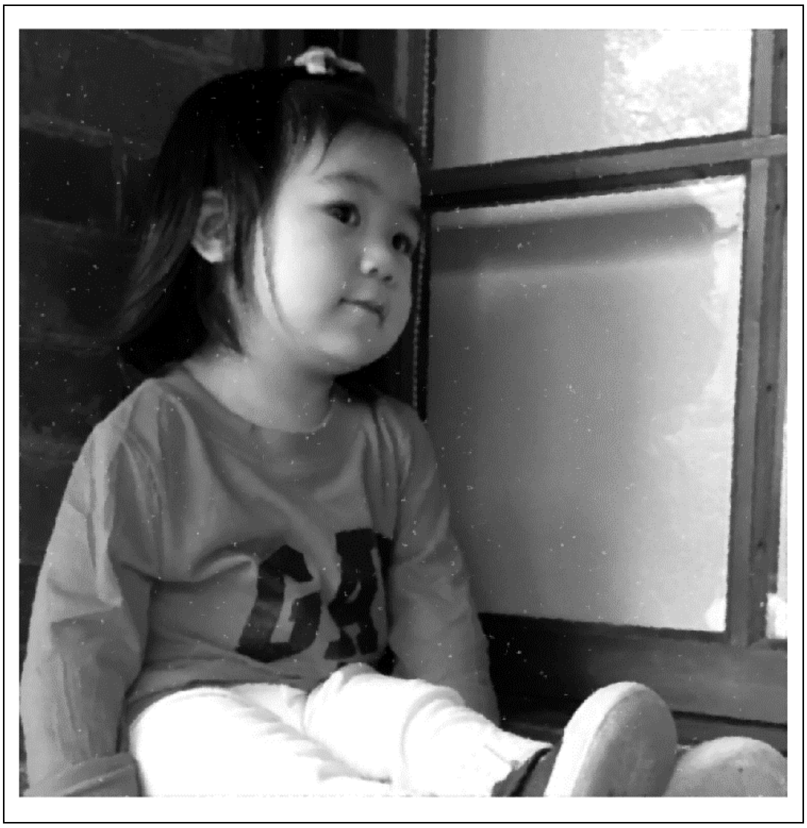
$\mathrm{P}_a = p_d(0) - p_0(0) = 0.0578 - 0.0081 = 0.049$

$\mathrm{P}_b = p_d(255) - p_0(255) = 0.1955 - 0.05501 = 0.1405$

Histogram of kid with noise	Histogram of kid after de-noise
	

c. De-noise image by alpha-trimmed mean filter

Apply alpha-trimmed mean filter, size = 5*5, alpha = 16



d. Output image

1. Choose parameters

✓ Gaussian low pass filter :

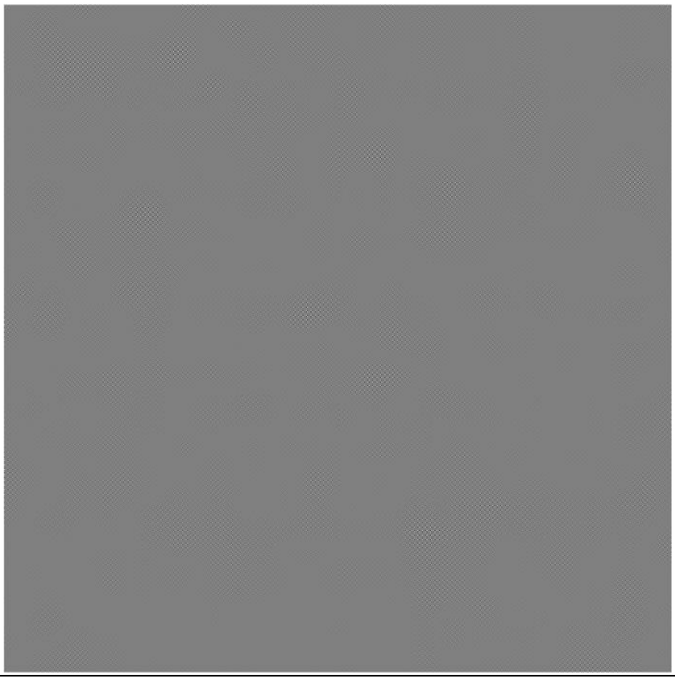
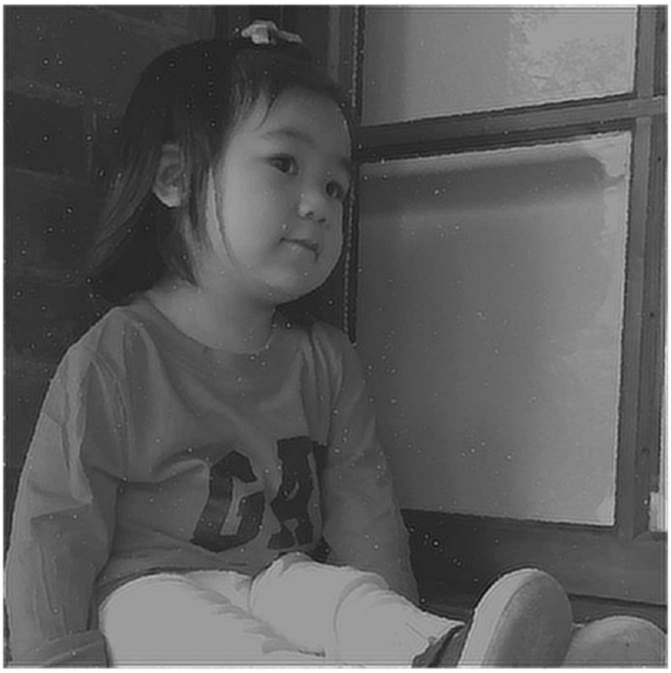
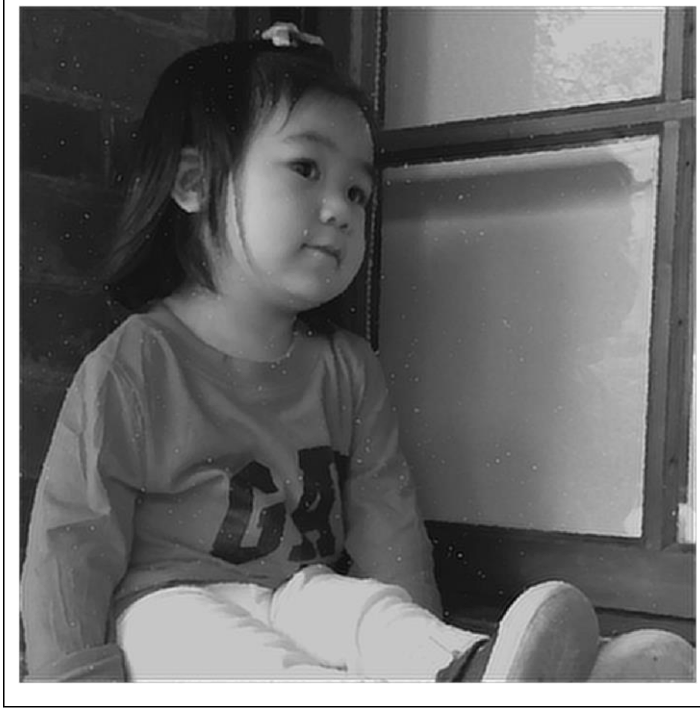
D0 = 250 (D0=250 has better output than D0=100 or 150 or 200)

✓ Butterworth low pass filter :

n = 10 (Choose much large n to make sure the sharp decrease in transition band)

D0 = 300 (choose by try and error, D0 can’t be too small but also can’t be too large)

2. Output image

D0 = 100 	D0 = 150 
D0 = 200 	D0 = 250