AMATH 482
Homework 3 Report
Fangzheng Sun
1239942

Abstract

In this project, I am going to recover some of the corrupted pictures of Derek Zoolander in black & White or RGB. I am going to use Gaussian filters and diffusion to remove the corrupted part of the pictures.

Introduction

Ugly protesters against the fashion industry sweat shops in Eurasia have corrupted Derek's photos. In the first two, noise are added on the whole pictures, and in the last two, only a small part on the face are corrupted. My job is to clear corruption and noise added to the pictures.

Theoretical background

The following theories are included in this project:

1. Fast Fourier Transform and Inverse Fast Fourier Transform. FFT converts a signal from its original domain (time/space) to a representation in the frequency domain. And iFFT does the inverse job.
2. Gaussian Filter. Given the center frequency, Gaussian filter is used in the frequency domain to help denoising the given frequencies by multiplying with the data in the frequency domain and results in a denoised frequency domain. In 2-D filtering, the Gaussian filter equation is:

$$F(k) = e^{-\tau(x-x_0)^2-\tau(y-y_0)^2}$$

Where τ measures the bandwidth of the filter and x0 and y0 are desired signals.

3. Diffusion. For denoising applications, or applications in which the data is choppy or highly pixelated, smoothing algorithms are of critical importance. One can take the original data as the initial conditions of a diffusion process with evolution $\partial u/\partial t = D(\nabla^2)u$ where $\nabla^2 = \partial x^2 + \partial y^2$. This diffusion process provides smoothing to the original data file u. The key is knowing when the smoothing process should be stopped so as not to remove too much image content.

Algorithm Implementation and Development

Task 1: Use filtering to try to clean up the corrupted images of Derek. Filter both the black and white image and the three levels of the RGB data of the color picture.

For black & white image, I first find the size of the image and locate its center (181, 127) to be the desired point of filtering. Then I converts it into double precision and FFT the image. Then I apply the filter to the image with 4 different sigma values, 0.01, 0.001, 0.0001 and 0, IFFT the filtered values back, converts the results into integer and show the filtered images with the four values of sigma.

For RGB image, the basic idea is similar to that of the black & color image. The only difference is that there are three layers of data in the image information, red, green and blue. I filter the three layers separately by the method introduced above and then combine them together in a denoised RGB image.

Task 2: Use diffusion to locally try to remove Derek's rash. You could potentially use both diffusion and filtering to get the best image quality. Again, you now have three diffusion equations to solve for the three levels of the color picture.

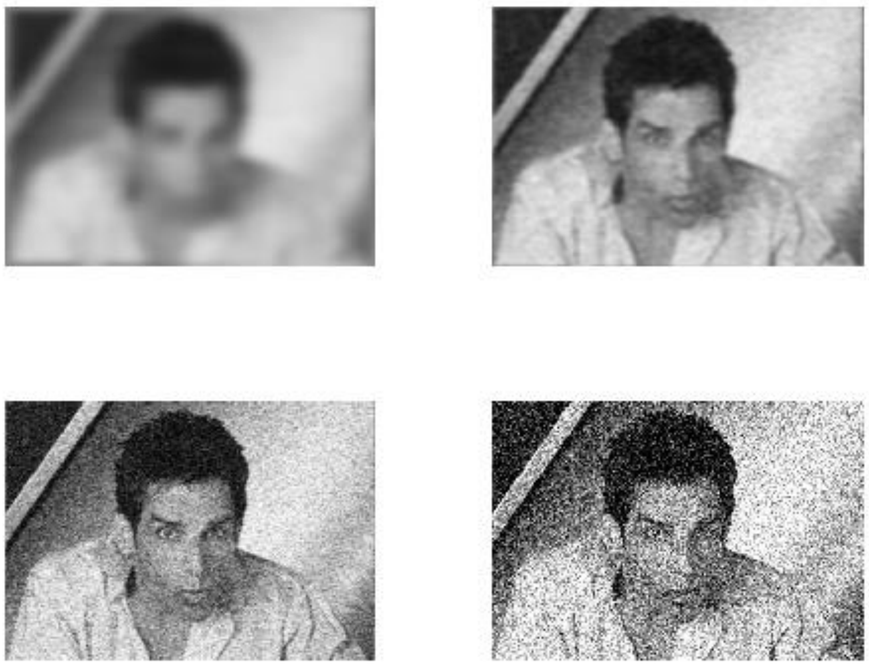
For black & white image, I first locate the corrupted area, around (140:160, 160:180). However, to clean this area up, I need to do diffusion on a wider area to make this area continuous to the surroundings. So I choose the area (130:170, 150:190) as my target. For the selected part of the image, I reshape the 41*41 information first. To make the Laplacian operator in two dimensions, I use the command kron and put the x- and y−derivatives in one dimension. The generated operator L takes two derivatives in x and y and is the Laplacian in two-dimensions. This operator is constructed to act upon data that has been stacked in one dimension. Then after several trials I determine a constant diffusion coefficient D, which is 0.2 and t as a function of time with 4 values, 0, 0.002, 0.004, 0.006. After that I apply ode113 and calls on the function image_rhs which contains the diffusion equation information to generate my desired information stack “Bsol”. Finally I reshape the stack back to 2 dimensions, converts to integers and replace the corrupted area (140:160, 160:180) with the same part in the adjusted image. The cleaned 4 images with different t values are generated.

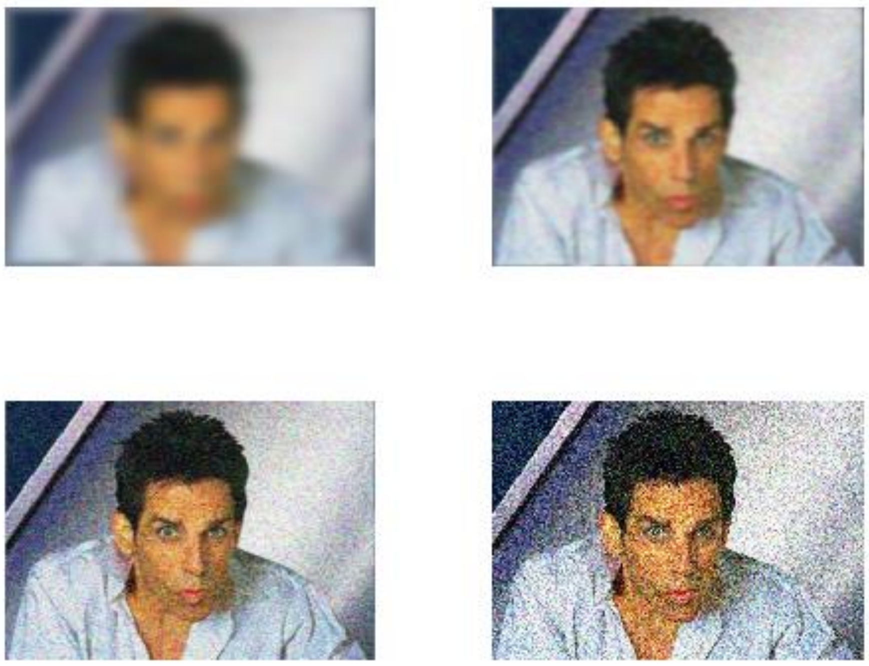
For RGB image, the basic idea is similar to that of the black & color image. The difference is that I use diffusion separately on the 3 layers and then combine them together in a RGB image.

Computational Results

Part 1:

The four pictures, are separately sigma = 0.01, sigma = 0.001, sigma = 0.0001 and sigma = 0 (Original picture)

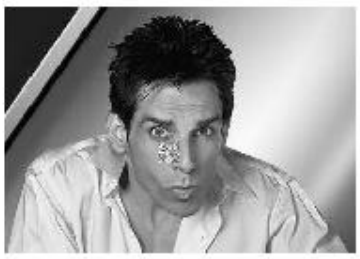


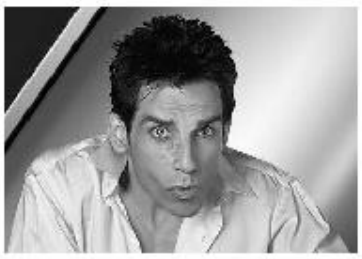


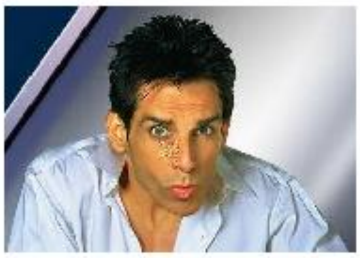
From the comparison, we find that when sigma = 0.0001, the picture is the best in the four.

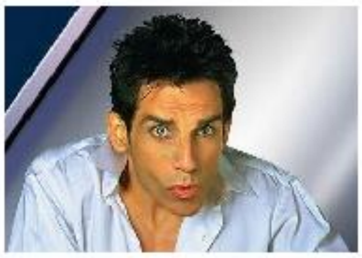
Part 2:

The four pictures, are separately t = 0 (original picture), 0.002, 0.004, 0.006 and the diffusion coefficient D = 0.2.









The picture 3 or 4 are pretty good, with t = 0.004 and t = 0.006.

Summary and conclusions

By using filtering and diffusion, I successfully clean the polluted pictures. Although they are not very perfect, the adjusted images are much better than the original polluted ones.

Appendix A: MATLAB functions

linspace – Generate linearly spaced vector

fft2 – 2-D Fast Fourier Transform

fftshift – Shift Zero frequency component to center of spectrum

ifft2 – 2-D Inverse Fast Fourier Transform

meshgrid – Create rectangular grid in 2-D space

reshape – reshape array to 2-D matrix

double – convert to double precision

uint8 – convert to 8-bit unsigned integer

ones – create array of all ones

spdiags – extract and create sparse band and diagonal matrices

kron – kronecker tensor product

ode113 - solve nonstiff differential equations; variable order method

Appendix B: MATLAB codes

Part 1 & Part 2

```
clear all; close all; clc

%% Part 1 B&W

Abw=imread('derek2','jpeg');
[ny,nx]=size(Abw);
x0=double(uint8(nx/2)); y0=double(uint8(ny/2));
Abw=double(Abw);
kx=1:nx; ky=1:ny;
[Kx,Ky]=meshgrid(kx,ky)
At=fft2(Abw);Ats=fftshift(At);

%Gaussian filter
fs=[0.01 0.001 0.0001 0];
figure(1)
for j=1:4
   F=exp(-fs(j)*(Kx-x0).^2-fs(j)*(Ky-y0).^2);
   Atsf=Ats.*F; Atf=ifftshift(Atsf); Af=ifft2(Atf);
   subplot(2,2,j), imshow(uint8(Af)), colormap(gray);
end

%% Part 2 RGB

A=imread('derek1','jpeg');
[ny,nx,nz]=size(A);
x0=double(uint8(nx/2)); y0=double(uint8(ny/2));
A=double(A);
kx=1:nx; ky=1:ny;
[Kx,Ky]=meshgrid(kx,ky)

%Gaussian filter
fs=[0.01 0.001 0.0001 0];
figure(2);
for j=1:4
   F=exp(-fs(j)*(Kx-x0).^2-fs(j)*(Ky-y0).^2);
   for k=1:nz
      At=fft2(A(:,:,k));Ats=fftshift(At);
      Atsf(:,:,k)=Ats.*F; Atf(:,:,k)=ifftshift(Atsf(:,:,k));
Af(:,:,k)=ifft2(Atf(:,:,k));
   end
   subplot(2,2,j), imshow(uint8(Af));
end

%% Part 2 B&W
```

```
Bbw=imread('derek4','jpeg');
B2=Bbw(130:170,150:190);
B2=double(B2);
[nx,ny]=size(B2);
B2_2=reshape(B2,nx*ny,1);

x=linspace(0,1,nx); y=linspace(0,1,ny);
dx=x(2)-x(1); dy=y(2)-y(1);
onex=ones(nx,1); oney=ones(ny,1);
Dx=(spdiags([onex -2*onex onex],[-1 0 1],nx,nx))/dx^2; Ix=eye(nx);
Dy=(spdiags([oney -2*oney oney],[-1 0 1],ny,ny))/dy^2; Iy=eye(ny);
L=kron(Iy,Dx)+kron(Dy,Ix);

tspan=[0 0.002 0.004 0.006]; D=0.2;
[t,Bsol]=ode113('image_rhs',tspan,B2_2,[],L,D);
for j=1:length(t)
   B2_clean=uint8(reshape(Bsol(j,:),nx,ny));
   Bbw(140:160,160:180)=B2_clean(11:31,11:31);
   subplot(2,2,j), imshow(Bbw);
end

%% Part 2 RGB

B=imread('derek3','jpeg');
B2=B(130:170,150:190,:);
B2=double(B2);
[nx,ny,nz]=size(B2);
for j=1:nz
   B2_2(:,:,j)=reshape(B2(:,:,j),nx*ny,1);

   x=linspace(0,1,nx); y=linspace(0,1,ny);
   dx=x(2)-x(1); dy=y(2)-y(1);
   onex=ones(nx,1); oney=ones(ny,1);
   Dx=(spdiags([onex -2*onex onex],[-1 0 1],nx,nx))/dx^2; Ix=eye(nx);
   Dy=(spdiags([oney -2*oney oney],[-1 0 1],ny,ny))/dy^2; Iy=eye(ny);
   L=kron(Iy,Dx)+kron(Dy,Ix);

   tspan=[0 0.002 0.004 0.006]; D=0.2;
   [t,Bsol]=ode113('image_rhs',tspan,B2_2(:,:,j),[],L,D);
   sol(:,:,j)=Bsol;
end
for j=1:length(t)
   for k=1:nz
```

```
        Bsol=sol(:,:,k);
        B2_clean(:,:,k)=uint8(reshape(Bsol(j,:),nx,ny));
    end
    B(140:160,160:180,:)=B2_clean(11:31,11:31,:);
    subplot(2,2,j), imshow(B);
end
```

function image_rhs

```
function rhs=image_rhs(t,u,dummy,L,D)
rhs=D*L*u;
```