CS6491 Topics in Optimization and its applications in CS

Final Report

Name: Yu Runzhou
Number: 55819400
Project Topic: 6 (Application of optimization in image denoising)
Software Link: https://github.com/rainyrubyzhou/CVX_project.git

1. Introduction

Digital image processing is a very popular technique with many applications. There're many tools that specialize in processing images, from professional software like photoshop to many easy-to-use built-in programs in mobile apps that can automatically improve photos. It's also the foundation of more complex techniques that solve computer vision problems like pattern recognition, video stream processing, and 3D modelling. Among topics in digital image processing, image denoising is an interesting one. In this report, I'll discuss the application of optimization in image denoising problem.

2. Background and problem statement

The digital images we can see in our daily lives can be generated from multiple sources: photos are taken by cameras, document copies are generated by scanners, medical images like X-ray images generated by specialized machines. During the process of generation, noises could be brought in by sensor or generated by the circuitry of the machine[1]. Sometime the noise could greatly influence the information we want to extract from the image or the content we want to express in the image. It's especially annoying when the main function of the image is affected. For example, noises can sometimes make it difficult for doctors to diagnose the disease from medical images. Therefore, it's necessary to denoise and enhance the quality of noised images.

Digital images are usually represented using matrixes, with each entry of the matrix indicating the intensity of the corresponding pixel in the picture. Values of matrix elements are in range [0, 255], with 0 representing black and 255 representing white. For colored images, matrixes of three tunnels RGB (red, green, blue) are used to store the grayscales of different color components separately. To simplify the problem model, we will focus on denoising of gray pictures, that is, single tunnel denoising problem.

Suppose the given noised image is Y, the original image is X, the noise denoted as N, we have $Y = X + N$. With only Y given, our goal is to calculate the $\hat{X}$, so than $\hat{X} - X$ is minimized.

3. Solution design

[1] https://en.wikipedia.org/wiki/Image_noise

As we have no idea what the noise N and original image X are, we can think about denoising in another way: on the one hand, we want the output $\hat{X}$ be based on the noised input, that is to minimize the distance between $\hat{X}$ and Y; on the other hand, we want the output $\hat{X}$ to fit the features of original X, and this is where we can design different strategies.
Now the problem has become a multi-objective optimization, but we can easily transform it into a single objective problem by weighted sum:

$$minimize\ \lambda \left\|\hat{X} - Y\right\|_2 + f(\hat{X})$$

and this is an unconstraint optimization problem.
To measure the distance between input and output, we can use L1 norm or L2 norm. To measure the feature function, we can first have a look at some common image noises:

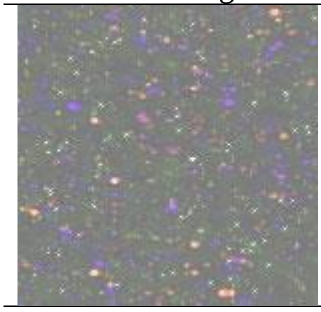
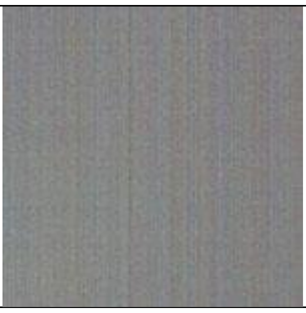
		
Fixed pattern noise	Random noise	Band noise

Table 1 Some common noises[2]

Intuitively, we can see that noises in images are frequent and small impulses, and they make the image less smooth. From this heuristic, we can use a metric of smoothness as the objective f.

Three methods have been implemented and tested, and we will name them f_1, f_2, f_3 respectively.

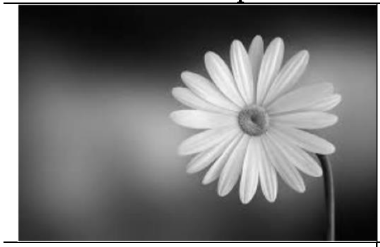
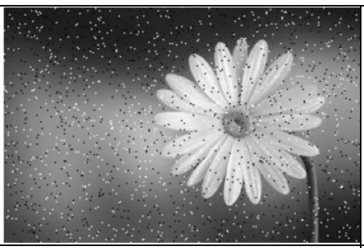
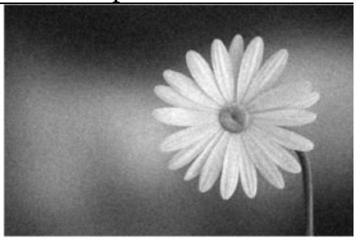
a. $minimize\ \lambda \left\|\hat{X} - Y\right\|_2 + f_1(\hat{X})$,
   With $f_1(X) = \sum \left\| \begin{bmatrix} X_{i+1,j} - X_{ij} \\ X_{i,j+1} - X_{ij} \end{bmatrix} \right\|_2$, which is implemented by cvxpy using tv(X).
b. $minimize\ \lambda \left\|\hat{X} - Y\right\|_2 + f_2(\hat{X})$
   With $f_2(X) = \|X[:,1:end-1] - X[:,2:end]\|_1 + \|X[1:end-1,:] - X[2:end,:]\|_1$, which is implemented by calculating the difference matrix of difference axis on $\hat{X}$ and then sum the L1 norm of both.
c. $minimize\ \lambda \left\|\hat{X} - Y\right\|_2 + f_3(\hat{X})$
   With $f_3(X) = \|X[:,1:end-1] - X[:,2:end]\|_2 + \|X[1:end-1,:] - X[2:end,:]\|_2$, which is implemented by calculating the difference matrix

[2] https://www.cambridgeincolour.com/tutorials/image-noise.htm

of difference axis on $\hat{X}$ and then sum the L2 norm of both. This is variation on 2D of quadratic smoothing on 1D data.

4. Examples and Analysis

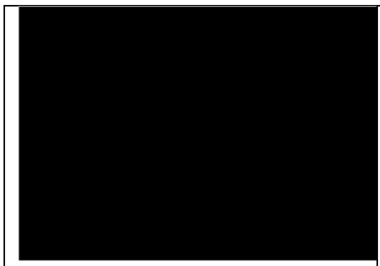
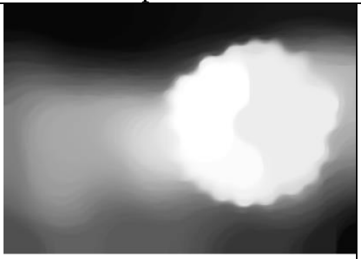
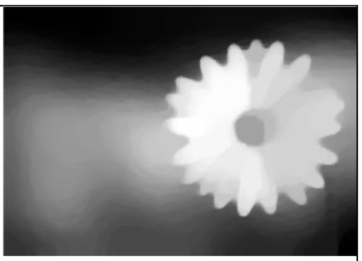
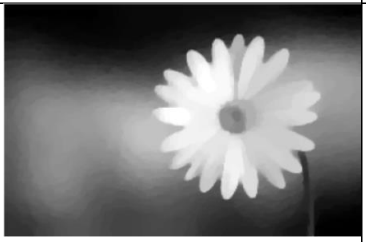
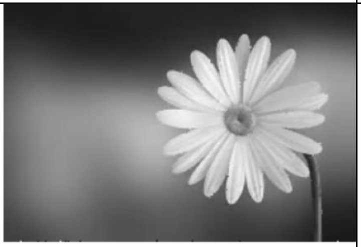
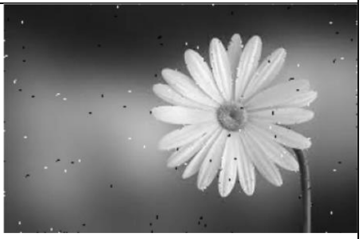
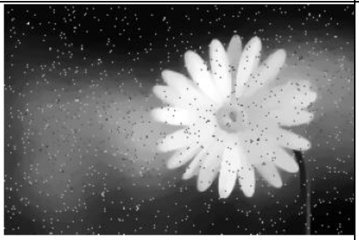
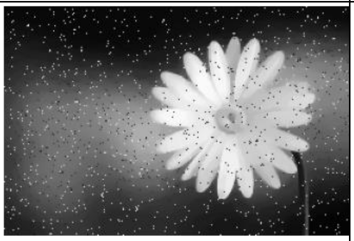
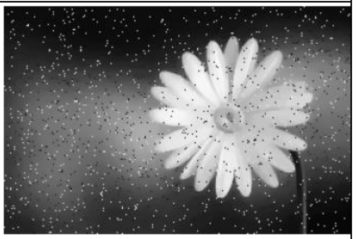
In this section we will use the image below as an example. We'll add two different noises (salt and pepper noise, gaussian noise) into the image. And then we'll present results of different methods and different parameters.

		
Original image	Salt & pepper noised	gaussian noised

4.1 Results of different metric of smoothness.

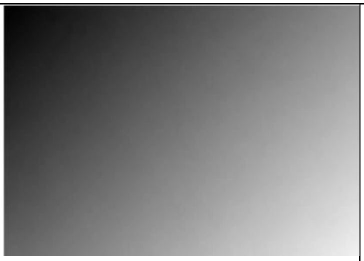
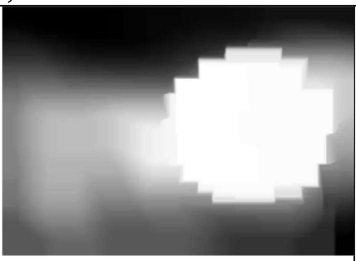
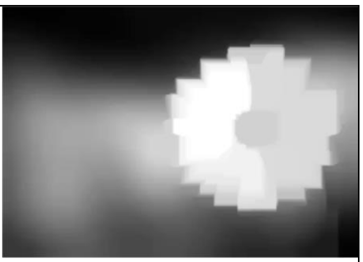
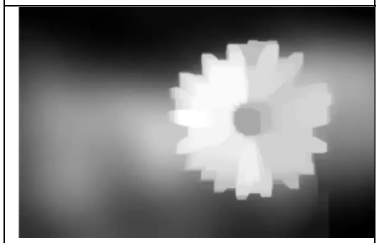
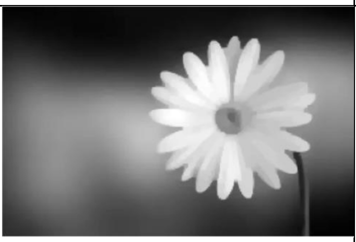
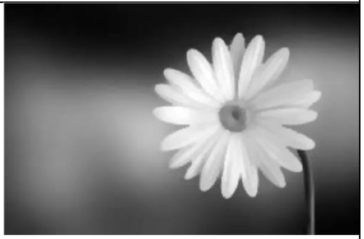
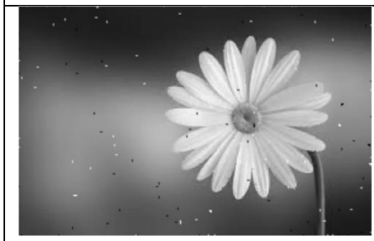
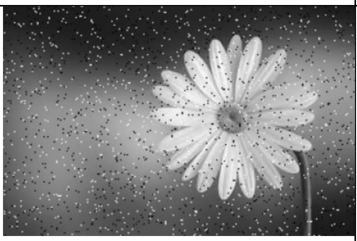
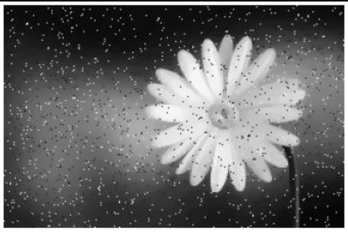
To make a trade-off between the distance with input image and the smoothness of output, different λ are tested to show the change of optimization result in each group. Ranges of λ are differently selected to show the change from totally smooth output to the exact same with input.

a. Using f_1.
Results in the table shows the outputs with λ from 0 to 10.

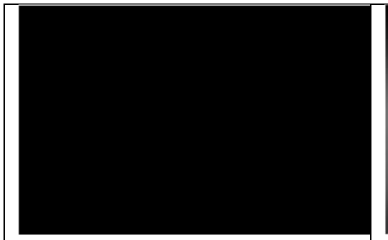
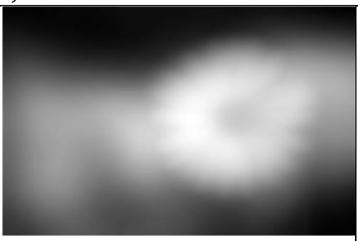
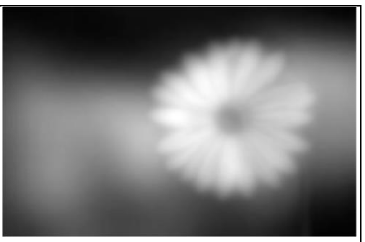
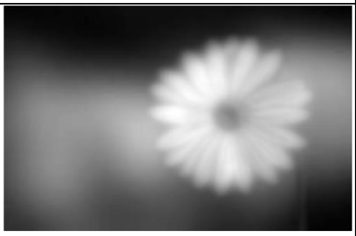
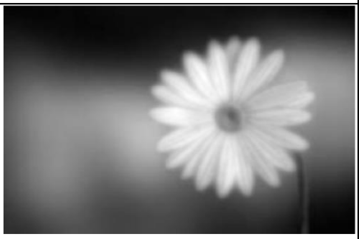
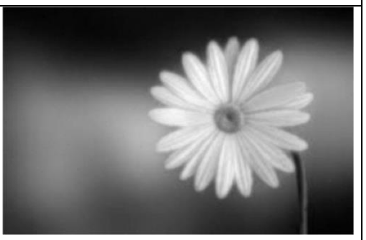
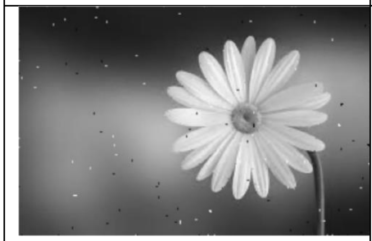
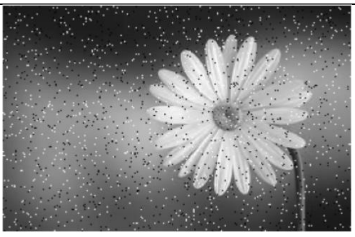
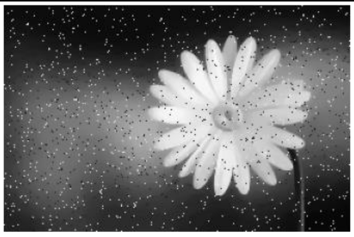
b. Using f_2.

For results in this table, λs are set from 0 to 10.

c. Using f_3.

For results in this table, λs are set from 0 to 0.2.

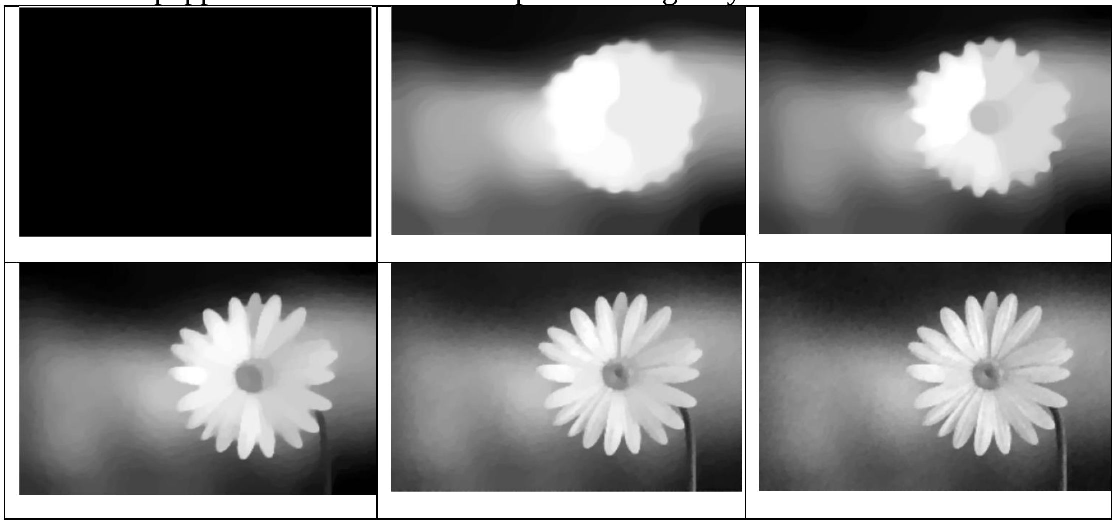
		
		
		

As we can see from three groups, changes of denoised pictures are quite different when using different smoothness measurements. Both f_1 and f_3 start

from a totally black image while f_2 starts with a certain gradient when the weight of distance part is 0.
At early stage, f_3 keeps the shape of the flower best while f_1 and f_2 output images with blurred contours. Shapes of the contours are relevant with the calculation of f.

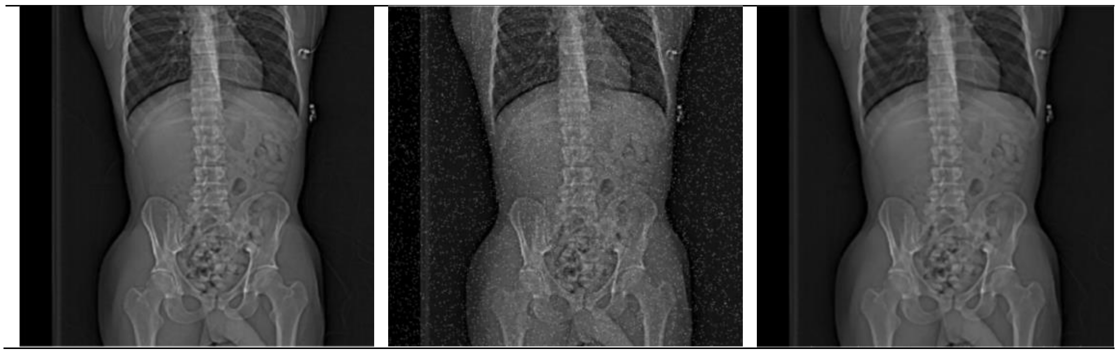
4.2 Denoising results on different noises.

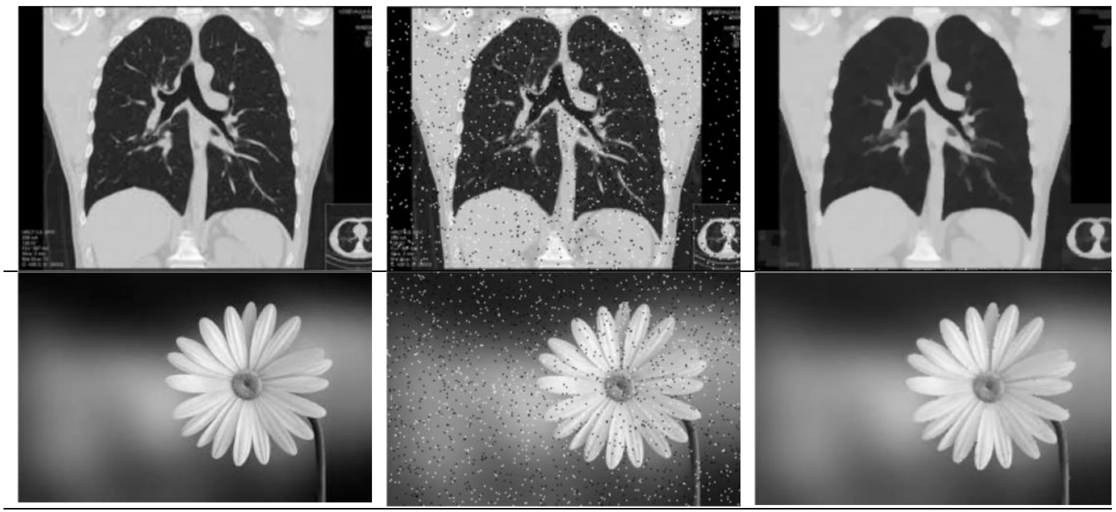
Figures below show the outputs for image with gaussian noise as λ changes. Compared with results in 4.1, the program gets better denoising results on salt and pepper noise because the impulses brought by noise are more obvious.



4.3 Denoising results on different images:

Because different images have different structures and features, to test the general utility of the algorithm, here we present three groups of experiments, with original image, noised image with salt and pepper noise ratio of 0.02, denoised image by minimizing the sum total variation and L1 norm of the difference.
In general, the algorithm works well with all three pictures. But the second picture with more details than the other two has been blurred and it’s hard to keep lossless while denoising.





Original image	Noised image	Denoised image

4.4 Running time analysis

For average running time, f_2 is faster because it uses L1 norm, while the other two are close.

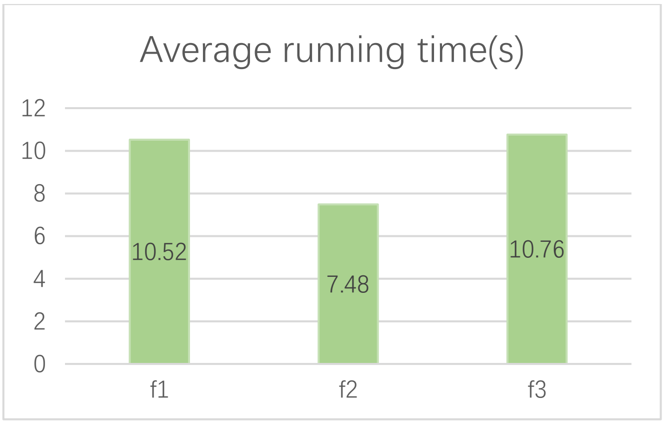


Table 2 Average running time

Also, the running time increases first and then decreases as λ changes from 0 to larger values.

5. Potential Improvements

As has been shown in previous part, each algorithm works differently on different type of noises. It’s better to design denoising operators based on the scheme of the generation of noises.
Also, the texture and feature of the main content in the image should also be considered. For example, when there’re too many details and minor contours in the original image, it would be hard to use the smoothness metric without losing some details. If the main part has clear contours and the background is clean, this type of methods will probably work well.