## Abstract

The purpose of this project is to investigate different optimization models and to utilize minimization methodologies that were introduced in the lecture to reconstruct images from partial data. In this project, firstly the FISTA and LBFGS method were used to repair pictures that showed with mask. Besides, several indicators such as PSNR were used to evaluate different parameters and methods. In addition, based on the parameters effect, meaningful analysis was acquired. Secondly, primal-dual method was used as an alternative way to inpainting those masked pictures again. Moreover, PSNR would also be used to measure different parameters in this method. Thirdly, a PDE-based image compression technique would be used to compress the nonconvex image appropriately.

## FISTA and L-BFGS method

### Project description

In this section, the purpose of the project is to repair images which could be 'polluted' by certain kinds of masks. Accordingly, there are two possible methods to achieve this goal. They are FISTA and L-BFGS. Firstly, by using FISTA method, following objective problem $\min\frac{1}{2}||Ax-b||^2+\mu||x||_1$ may be seen as the optimal object. In other words, $||Ax-b||^2$ means the difference between repaired part $Ax$ and the non-masked part in initial image $b$. while $\mu||x||_1$ represents the a very sparse representation and many of the components xi are zero or close to zero in frequency space. Secondly, by using L-BFGS method, following objective function $\min\frac{1}{2}||Ax-b||^2+\mu\sum_{i=1}^{mn}\varphi_{hub}(x_i)$ , may be seen as another optimal object, which $\sum_{i=1}^{mn}\varphi_{hub}(x_i)$ is a smooth part of objective function. This part could enable L-BFGS to be effective.

### Implementation description

Accelerated Proximal Gradient Method (FISTA) is a powerful tool to improve the performance and complexity results of the basic (proximal) gradient method. Figure 1 shows that $x_{k+1}$ which is DCT of inpainting picture, $y$ which is the estimator of $x_{k+1}$.Together with, using

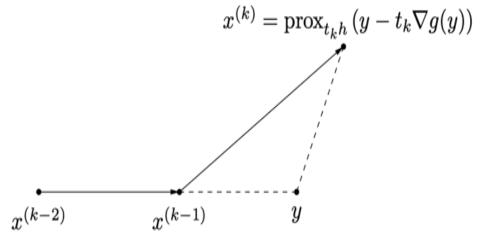
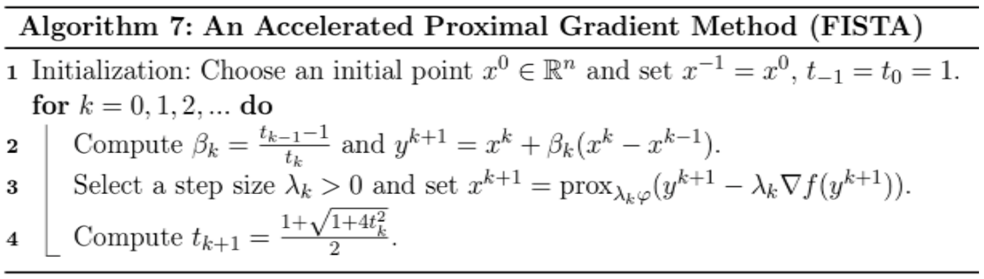


Figure 1.

backtracking method, the step size( $t$ ) was calculated. Moreover, as the formulation shows in picture the iterate steps were figured. Together with, L-BFGS method derives from BFGS, which is a quasi-newton method, it simulates the Hessian with an appropriate matrix to simplify difficulty of calculation. This method was achieved by storing the last several pairs of vectors to construct a much easy 'Hessian'. Figure 2 and 3 are algorithms of both methods.



```
Algorithm 7: An Accelerated Proximal Gradient Method (FISTA)
1  Initialization: Choose an initial point $x^0 \in \mathbb{R}^n$ and set $x^{-1} = x^0$, $t_{-1} = t_0 = 1$.
   for $k = 0, 1, 2, \ldots$ do
2      Compute $\beta_k = \frac{t_{k-1}-1}{t_k}$ and $y^{k+1} = x^k + \beta_k(x^k - x^{k-1})$.
3      Select a step size $\lambda_k > 0$ and set $x^{k+1} = \mathrm{prox}_{\lambda_k\varphi}(y^{k+1} - \lambda_k \nabla f(y^{k+1}))$.
4      Compute $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$.
```

Figure 2 source: lecture note of prof. Andre Milzarek Manfred

L-BFGS Algorithm Step:

1. Choose initial point $x_0$, tolerance $\varepsilon > 0$, k=0 and r = $\nabla f(x_0)$
2. If $\|\nabla f(x_k)\| \le \varepsilon$, return x. Otherwise, step into step 3.
3. Calculate step size $\alpha > 0$ by backtracking method. Condition is:

   $$f(x_{k+1}) - f(x_k) > -\alpha \times \gamma \times \nabla f(x_{k+1})^T \times r.$$

   Get $x_{k+1} = x_k - \alpha \times r$
4. Calculate $s_k = x_{k+1} - x_k$, $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$, $\rho_k = \frac{1}{s_k{}^T \times y_k}$
5. If Store $s_k{}^T \times y_k > 10^{-14}$, store $s_k, y_k, \rho_k$. If k > m, delete $s_{k-m}$, $y_{k-m}$, $\rho_{k-m}$
6. Using two-loop recursion to calculate $r$. $H_k^0 = \frac{s_{k-1}{}^T \times y_{k-1}}{\|y_{k-1}\|^2} \times I$ .

```
$q \leftarrow \nabla f_k$;
for $i = k-1, k-2, \ldots, k-m$
    $\alpha_i \leftarrow \rho_i s_i^T q$;
    $q \leftarrow q - \alpha_i y_i$;
end (for)
$r \leftarrow H_k^0 q$;
for $i = k-m, k-m+1, \ldots, k-1$
    $\beta \leftarrow \rho_i y_i^T r$;
    $r \leftarrow r + s_i(\alpha_i - \beta)$
end (for)
```

7. $k = k + 1$, then step into step 2.

Figure 3 L-BFGS Algorithm Step

**Result and observation**

In this process, firstly the result for two methods were showed, then different parameters were adjusted to explore whether there could be better way to execute this experiment.

**Result**

According to the algorithm described above, the matlab code is followed. After the code implementation is completed, both grey and color images' repair are implemented, as shown in image 1-4 below. In this section, the tested samples used are lion in grey and color version. It is noted that for color images FISTA algorithm and LBFGS algorithm are still applicable. Different from gray image, image reconstruction needs to be carried out on the three colors channels of RGB

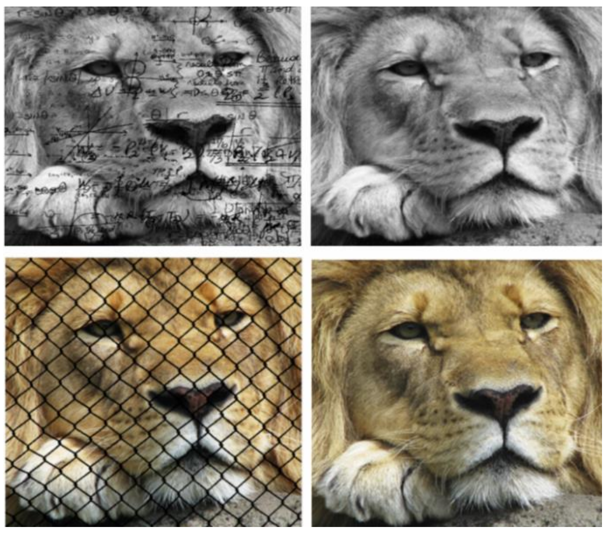


Image 1-4: handwriting, recoverd_handwriting, mesh, recovered_mesh

**Observation**

· **Comparation in L-BFGS**

After verifying the effect of the code, the suitable parameters will be found. Because of L-BFGS algorithm and Huber-function, number of storage $m$ and $\delta$ are changeable parameters. Thus, firstly the $\mu$ and $m$ were fixed and adjusting $\delta$ to see whether the other indicators such as Iteration Number and PSNR change. Thus, following table 1 shows that the adjustment of $\delta$ only affects the execution time, not the number of iterations and the PSNR value. Together with, the $\mu$ and $\delta$ were fixed and adjusting $m$ to see whether the other indicators varies. Therefore, table 2 indicates that the adjustment of $m$ only affects the iteration steps and execution time, not the final PSNR value

Table 1: $\delta$ effect with $\mu = 0.01$, $m = 5$

| $\delta$ | Iteration Number | CPU Time(sec) | PSNR |
|---|---|---|---|
| 0.01 | 103 | 23.68 | 31.14 |
| 0.05 | 103 | 21.21 | 31.14 |
| 0.1 | 103 | 22.19 | 31.14 |
| 0.5 | 103 | 21.71 | 31.14 |

Table 2: $m$ effect with $\mu = 0.01$, $\delta = 0.1$

| $m$ | Iteration Number | CPU Time(sec) | PSNR |
|---|---|---|---|
| 5 | 103 | 22.19 | 31.14 |
| 7 | 104 | 23.94 | 31.14 |
| 10 | 104 | 27.44 | 31.14 |
| 12 | 107 | 29.37 | 31.14 |

· **Comparation between FISTA and L-BFGS**

In this comparation, lion(grey) and handwriting pictures were selected to be samples. Besides, this section explores the effect of $\mu$, mask and tolerance. It could be seen that with the same $\mu$, the Iteration Number and CPU Time(sec) were less in L-BGFS method. Thus, L-BGFS could be better in this situation. Pictures of line charts in appendix shows the convergence rate. It could be seen that both of these two methods convergent quickly.

Table 3: $\mu$ effect in FISTA

| $\mu$ | Iteration Number | CPU Time(sec) | PSNR |
|---|---|---|---|
| 0.001 | 816 | 31.42 | 30.39 |
| 0.005 | 816 | 56.13 | 30.39 |
| 0.01 | 816 | 61.83 | 30.39 |
| 0.05 | 816 | 62.39 | 30.39 |

Table 4: $\mu$ effect in L-BFGS with $\delta = 0.1$, $m = 5$

| $\mu$ | Iteration Number | CPU Time(sec) | PSNR |
|---|---|---|---|
| 0.001 | 103 | 21.10 | 31.14 |
| 0.005 | 103 | 21.30 | 31.14 |
| 0.01 | 103 | 21.05 | 31.14 |
| 0.05 | 103 | 21.48 | 31.14 |

- **Different Images with Different Details**

In order to compare the restoration effects of images with different detail level, Fast Fourier Transform is applied to the image firstly, which could reveal the detail level of an image. Image center in the right column of figure 1 represent low frequency, the more it diffuses around, the higher the frequency is. Here, we choose three images with different degrees of detail for reconstruction. It is shown that the more detail a graph has, the more high-frequency components it represents.

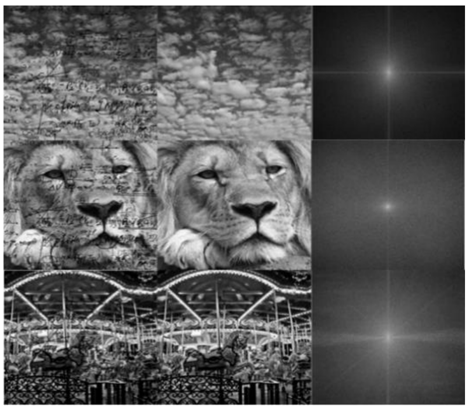


image 5-13: left: image 'clouds'; middle: image 'lion'; right: image 'carousel'

The following figure 2 displays the comparative PSNR curves among the three images mentioned above from left to right. It could be seen that the PSNR values in turn from about 40 to 20, which indicates that regardless of FISTA and L-BFGS, the less detailed the graph, the fewer steps the algorithm needs to perform; The more detail the graph has, he more steps the algorithm needs to perform.

In addition, the iteration times could be considered as a criterion to measure the difficulty degree of image reconstruction. Figure 3 depicts the convergence process for the three sets of images. From the result, the stopping steps of image 'clouds' is the lowest, while image 'carousel' has the most iteration times. Therefor image with relatively high level of detail may lead to quicker convergence, and vice versa. Table 5 and table 6 show this phenomenon more specificly.

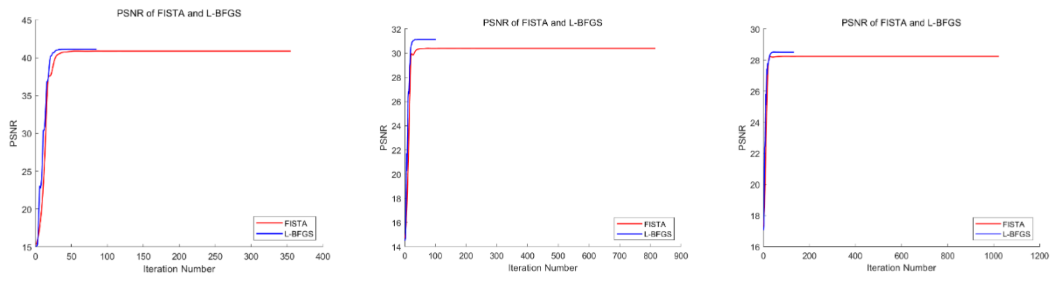


Figure 2

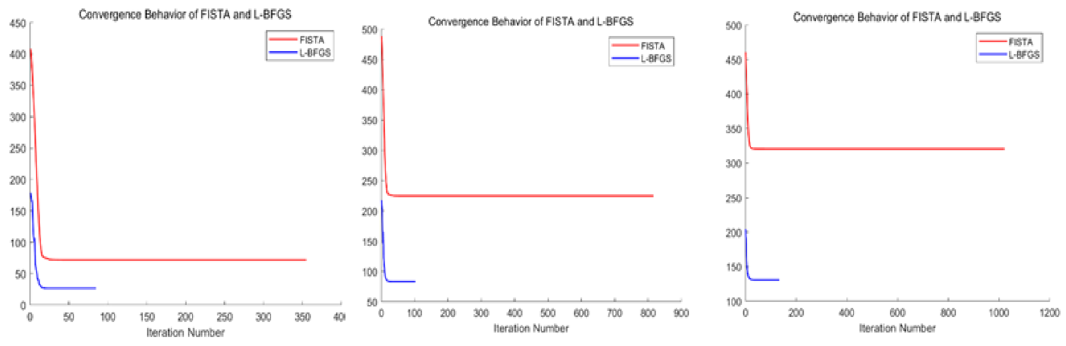


Figure 3

Table 5: detail effect in FISTA with $\mu = 0.01$

| Image | Iteration | CPU Time(sec) | PSNR | Details |
|---|---|---|---|---|
| clouds | 355 | 22.92 | 40.88 | low |
| lion_grey | 816 | 54.39 | 30.39 | medium |
| carousel | 1022 | 77.11 | 28.24 | high |

Table 6: detail effect in with L-BFGS with $\mu = 0.01$, $\delta = 0.1$

| Image | Iteration | CPU Time(sec) | PSNR | Details |
|---|---|---|---|---|
| clouds | 86 | 17.13 | 41.12 | low |
| lion_grey | 103 | 22.19 | 31.14 | medium |
| carousel | 134 | 28.39 | 28.51 | high |

- **Lion images with different Mask**

In this part, different mask rates were tested to figure out whether the mask rate affect the repair effect. As images showed, the left column represents the rate of 50%, 70% and 90%. Thus, it is obvious that with the increase of mask rate, the repaired effect became lower which present in the second column, compared with the initial images in the last column. Together with, from the tables below, the higher the mask rate, the more iterations and time were needed.

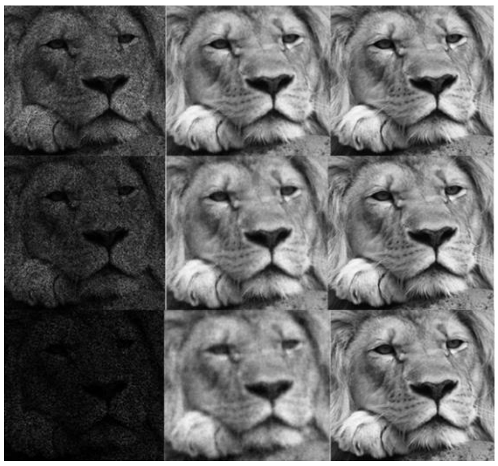


Image 14-22

Table 7: mask effect in with FISTA with $\mu = 0.01$ $\delta = 0.1$

| Mask rate | Iteration Number | CPU Time(sec) | PSNR |
|---|---|---|---|
| 0.5 | 1409 | 77.11 | 24.89 |
| 0.7 | 2213 | 151.82 | 22.51 |
| 0.9 | 4095 | 277.54 | 20.11 |

Table 8: mask effect in with L-BFGS with $\mu = 0.01 \quad \delta = 0.1 \quad m = 5$

| Mask rate | Iteration Number | CPU Time(sec) | PSNR |
|---|---|---|---|
| 0.5 | 103 | 27.86 | 25.46 |
| 0.7 | 141 | 28.50 | 23.10 |
| 0.9 | 207 | 41.71 | 20.48 |

**Different tolerance**

After the above parameters' adjustment, it could be found that the results of image restoration are only related to the original image itself and the destruction degree, for example, if the algorithm need perform over 1000 steps, the first one hundred steps have achieved the final result. Therefore, it could be supposed that, to a large extent, decision threshold can be adjusted at the end of the iteration algorithm is relatively easy, but still can get ideal result. Then the test began. After the adjustment of tolerance, conjecture could be well verified. In addition, as can be seen from the last two figures, FISTA and L-BFGS algorithms only need 20 steps to achieve minimizer, that is, the maximum value of PSNR. Knowing this feature, we do not need to set too large tolerance, which can save time.

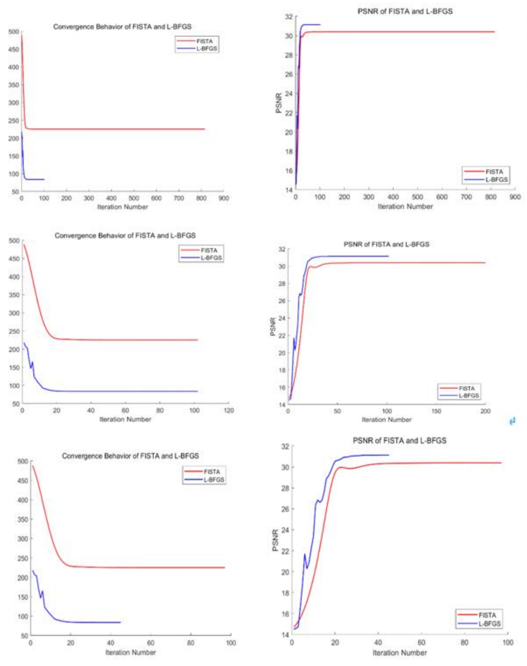


Figure 4

Table 9: mask effect in with FISTA with $\mu = 0.01$

| Tolerance | Iteration Number | CPU Time(sec) | PSNR |
|---|---|---|---|
| $10^{-6}$ | 816 | 31.42 | 30.39 |
| $10^{-5}$ | 560 | 29.62 | 30.39 |
| $10^{-4}$ | 357 | 27.84 | 30.39 |
| $10^{-3}$ | 195 | 14.90 | 30.39 |
| $10^{-2}$ | 97 | 7.20 | 30.39 |
| $10^{-1}$ | 49 | 3.69 | 30.36 |

Table 10: mask effect in with FISTA with $\mu = 0.01$ $\delta = 0.1$ $m = 5$

| Tolerance | Iteration Number | CPU Time(sec) | PSNR |
|---|---|---|---|
| $z$ | 103 | 21.10 | 31.14 |
| $10^{-5}$ | 89 | 20.98 | 31.14 |
| $10^{-4}$ | 76 | 16.78 | 31.14 |
| $10^{-3}$ | 60 | 13.39 | 31.14 |
| $10^{-2}$ | 45 | 9.66 | 31.14 |
| $10^{-1}$ | 30 | 6.41 | 31.07 |