

---

# Optimization for Total-Variation Image Denoising

---

Bingcheng HU

Department of Electric and Computer Engineering

Shanghai Jiao Tong University

Student ID: 516021910219

bingcheng@sjtu.edu.cn

## Abstract

Nowadays, smartphones are generally equipped with high-definition cameras, and the effect of taking pictures at night is always unsatisfactory, because when the light is weak, the internal and external noises become relatively large. TV regularization is an algorithm that reduces data noise by processing data in different dimensions. In this paper, the optimization of the TV denoising algorithm for monochrome pictures is discussed.

Keywords: total variation, regularized learning, sparsity, convex optimization

## 1 Introduction

Total variation has a long history in the field of image processing, and it was originally a good solution for signal denoising problems [Condat, 2013, Durand and Froment, 2003, Easley et al., 2008, Selesnick et al., 2014]. Condat [2013] propose a fast denoising algorithm for filtering discrete signals using total variation regularization [Condat, 2013]. The optimization problem of the one-dimensional (1D) discrete signals denoising is shown as below.

$$\min_{x \in \mathbb{R}^N} \frac{1}{2} \sum_{k=1}^N |y[k] - x[k]|^2 + \lambda \sum_{k=1}^{N-1} |x[k+1] - x[k]|$$

where  $y$  is the noised signal and by solving this optimization problem, we can get the denoised signal  $x$ . Condat [2013] presents an efficient way to implement this convex optimization problem, and one example of the usage of the algorithm in signal processing is shown in figure 1 [Condat, 2013].



Figure 1: One example of the noised signal (in red), the unknown ground truth (in green), and the TV-denoised signal  $x$  (in blue) [Condat, 2013].

12 Apart from the signal denoising, the total variance method was considered a well-established way to  
13 solve image processing problems [Chen et al., 2010, Blomgren et al., 1997, Chen et al., 2015, Thanh  
14 and Dvoenko, 2015]. In real life, people need to take photos at night and their phones always catch a  
15 noised picture because of the lack of light. Nowadays, many smartphone companies introduced lots  
16 of algorithms to improve the quality of the photo (Chen et al., 2010, Blomgren et al., 1997, Chen  
17 et al., 2015). In those algorithms, the total variant is still a very important part of denoising.

18 In the photograph field, Total variant denoising can help the photographer get a better picture. But for  
19 non-professionals, it is difficult to adjust the regularization parameter. Chen et al. [2010] proposed  
20 a majorization-minimization approach for adaptative total variation image denoising. Using their  
21 methods, regularization parameter doesn't need to be specified by authors. This layman-friendly  
22 method can easily make total variant image denoising expand into daily life. Figure 2 shows the  
23 practical application of this algorithm. The picture shows a picture of a zebra. The picture on the left  
24 is the original picture, and the picture with noise added in the middle. On the right is the image after  
25 noise reduction using the total variant denoising. By carefully comparing Figure 1.c and Figure 1.a,  
26 you can find that the denoised image has an excellent performance in detail retention.

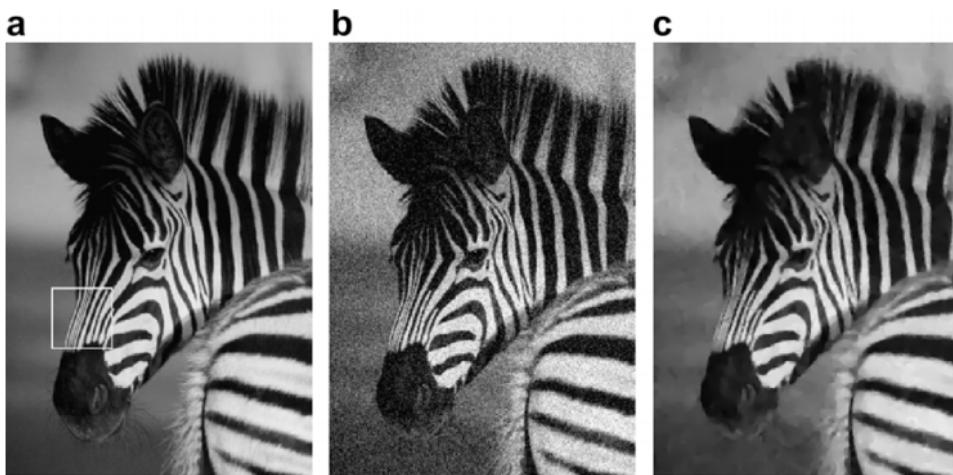


Figure 2: One example of the original image (a) , the noised image (b), and the TV-denoised image (c) [Chen et al., 2010].

27 Total variant image denoising is also a significant part of medical imaging. Thanh and Dvoenko  
28 [2015] proposes a TV denoising algorithm for biomedical images. This algorithm achieves noise  
29 reduction in line with medical standards by adjusting parameters. This noise reduction method will  
30 retain important details such as blood vessels, so that the details will not be erased by the noise  
31 reduction algorithm. However, this algorithm will cause color distortion, so it can not be used to  
32 denoise everyday photos. Figure 3 shows the MRI pictures of the human brain before and after noise  
33 reduction. Observing the pictures shows that the details of the images are well preserved.

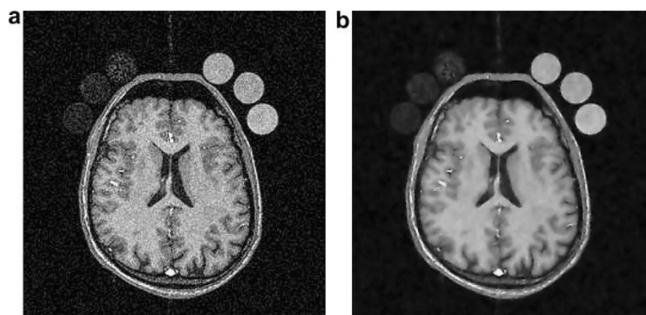


Figure 3: One example of the original noised image of the human brain scanned image (a) , and the TV-denoised image (b) [Thanh and Dvoenko, 2015].

In the field of total variant denoising, lots of new technologies were introduced these years. Many of them can achieve high quality, while still some algorithms were too slow for use in daily life []. Blomgren et al. [1997] studies one of the most useful images restoring algorithm and the optimization illustration of its problem is shown as below [Blomgren et al., 1997].

$$\min_u \alpha TV(u) + \frac{1}{2} \|\mathbb{K}u - z\|_{\mathcal{L}^2}^2$$

- 34 The Total Variation norm chosen in this equation is  $TV(u) = \int_{\Omega} |\nabla u| dx dy$ . Because the Total  
 35 Variation norm does not penalize disruption in  $u$ , it ensures a better restoration for edges[Blomgren  
 36 et al., 1997].

Beck and Teboulle [2009] introduced gradient-based strategies for image denoising and deblurring problems, and the model of the algorithm is based on TV minimization model with constraints [Beck and Teboulle, 2009]. The innovative invention of this paper is the usage of a fast iterative shrinkage/thresholding algorithm (FISTA) with its "novel monotone version" [Beck and Teboulle, 2009]. The convex non-smooth minimization problem is represented as below.

$$\min_{\mathbf{x}} \|\mathcal{A}(\mathbf{x}) - \mathbf{b}\|^2 + 2\lambda TV(\mathbf{x}), \quad (\lambda > 0)$$

Where  $TV(\Delta)$  stands for the l1-based, anisotropic TV, which is defined by

$$\begin{aligned} \mathbf{x} \in \mathbb{R}^{m \times n}, \quad TV_{l_1}(\mathbf{x}) = & \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \{|x_{i,j} - x_{i+1,j}| + |x_{i,j} - x_{i,j+1}|\} \\ & + \sum_{i=1}^{m-1} |x_{i,n} - x_{i+1,n}| + \sum_{j=1}^{n-1} |x_{m,j} - x_{m,j+1}| \end{aligned}$$

- 37 which is more complex than we have learned in class.

Chen et al. [2015] invented the fractional-order TV denoising model. The given image is denoted as the function  $f : \Omega \rightarrow \mathbb{R}$ , where  $\Omega$  is the subset of  $R^2$  in the image domain and the fractional-order TV denoising model is described by

$$\min_u \int_{\Omega} \frac{1}{2} (u - f)^2 + \mu |\nabla u| d\Omega,$$

- 38 and after minimize this function,  $u$  will be the clean image which is wanted.

But because of the non-differentiability of the fractional-order TV regularization term, this paper used a proximity algorithm to solve it, which is a discrete model formulation

$$\min_p \frac{1}{2} \|p - g\|_2^2 + \mu \|A^\alpha p\|_1$$

- 39 where matrix  $A^\alpha = [A_1^\alpha, A_2^\alpha, \dots, A_N^\alpha]^T \in \mathbb{R}^{2N \times N}$ .

Whatever the methods used in those papers, the regularization term is always the key point in controlling the model complexity. In this case, the least absolute shrinkage and selection operator(LASSO) needs to be introduced as figure 4 shows.

$$\min_x \gamma \sum_{j=1}^n |x_j| + \frac{1}{2} \sum_{i=1}^m (x^T a_i - b_i)^2$$

In this formula, both the l1 norm and the l2 norm can be used. But using the l1 norm can keep more sparsity. First, let's check the l2 norm

$$\min_x \gamma|x| + 1/2(x - 1)^2$$

- 40 where we can get  $f'(x) = \gamma \operatorname{sgn}(x) + (x - 1)$  such that  $x^* = 0$  if  $\gamma > 1$ .

And for l1 norm

$$\min_x 1/2\gamma x^2 + 1/2(x - 1)^2$$

- 41 where we can get  $f'(x) = \gamma x + (x - 1)$  such that  $x^* \neq 0$ .

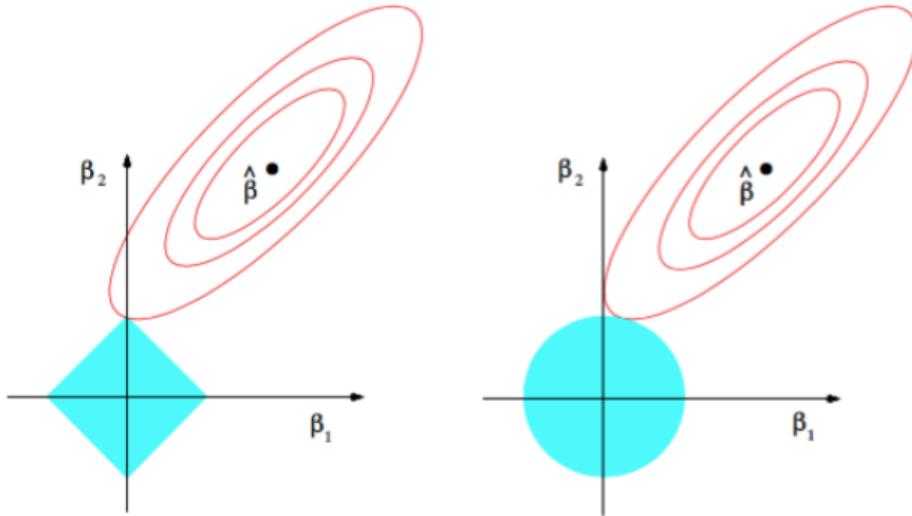


Figure 4: Estimation picture of the lasso (left) and ridge regression (right) [Friedman et al., 2001].

42 Both L1 norm and L2 norm regularization can help reduce the risk of overfitting, but the former also  
 43 brings an additional benefit: it is easier to obtain "sparse" (sparse) solutions than the latter, that is,  
 44 what it finds  $\omega$  will have fewer non-zero components. This characteristic is marked as the following  
 45 picture:

46 The previous section introduced some literature that used different Total Variance, and also introduced  
 47 some basic knowledge about convex optimization. The following section will introduce the problems  
 48 we need to solve.

## 49 2 Problem Statement

50 The goal of this article is to implement a real TV denoising algorithm from the python code step by  
 51 step. For any picture, it can be shown as  $X \in R_{n \times n}$ . However, sparsity is not an inherent property of  
 52 pictures, so we need a way to express the sparsity of pictures. Obviously, the difference between a  
 53 pixel in the picture and the surrounding pixels can indicate whether this pixel is conspicuous in the  
 54 picture. However, the noise of the picture will often produce many conspicuous points, which will  
 55 affect the clarity of the picture.

Therefore, in order to eliminate noise, we need to propose a method to represent the number and intensity of the noise point in the image. This method is using the total variation (TV). For picture X, its TV is written as below.

$$\|X\|_{TV} = \sum_{i=1}^{n-1} \sum_{j=1}^{n-1} \sqrt{(X_{i,j} - X_{i+1,j})^2 + (X_{i,j} - X_{i,j+1})^2} \text{ or } \|X\|_{TV} = \sum_{i=1}^{n-1} \sum_{j=1}^{n-1} |x_{i,j} - X_{i+1,j}| + |X_{i,j} - X_{i,j+1}|$$

56 Which of these two formulas is better? The two adjacent stores in the first formula are coupled  
 57 together, which means that these two items cannot be handled separately. This leads to optimization  
 58 difficulties. However, the two terms in the second formula are separate, which means that the two  
 59 terms can be handled separately. From this perspective, the second method is more convenient.  
 60 However, in fact, the adjacent points in the picture are coupled together, so using the first method will  
 61 get better results. So we decided to use the left formula as part of our TV denoising algorithm.

62 Could it be possible that reducing the total variation help reduce noise? Through the above analysis,  
 63 we know that the answer is yes. But if you do not set a limit, and directly reduce the total variation, it  
 64 will cause other hard problems, such as the required details are eliminated and the sharp edges of the  
 65 original become blurred. So in our optimization process, we will need an item so that the difference  
 66 between denoised image and Noisy image is not too large. So a new variable is introduced to this  
 67 problem.

$$dissimilarity = \|F - X\|_2^2$$

- 68 By minimize this term, we can make the denoised image and Noisy image similar to each other.  
 69 Finally, our optimization problem becomes the following form:

$$\min_X \lambda \|X\|_{TV} + \|F - X\|_2^2$$

- 70 In the following part of this article, the main task is to achieve an optimized solution to this problem  
 71 through Python programming.

### 72 3 Total Variance in Image denoising

- 73 In this section, the complete derivation process of the image denoising algorithm will be introduced.  
 74 To make the start simpler to understand, we choose the general gradient descent (GD) algorithm as  
 75 the first step of denoising.  
 76 As stated in the previous section, our aim is to

$$\min_X \lambda \|X\|_{TV} + \|F - X\|_2^2.$$

- 77 According to the gradient descent algorithm, the problem can be solved with the following procedures:

---

#### Algorithm 1 Gradient Descent Algorithm

---

```

procedure GRADIENT DESCENT ALGORITHM WITH EXACT SEARCH
  given a starting point  $x \in \text{dom } f$ 
  repeat:
    1.  $\Delta x := -\nabla f(x)$ 
    2. Line search. Choose step size  $t$  via exact or backtracking line search.
    3. Update.  $x := x + t\Delta x$ 
  until stopping criterion is satisfied.

```

---

- 78 So it's needed to compute  $\Delta x := -\nabla f(x)$  every step, then we need to do exact line search in the  
 79 direction of the negative side of the gradient.

#### 80 3.1 Calculate the gradient of the aim function

- 81 In this section, the procedure of calculating the partial differentiation. Both of the following two  
 82 items will be partially differentiated separately.

$$\|X\|_{TV} = \sum_{i=1}^{n-1} \sum_{j=1}^{n-1} \sqrt{(X_{i,j} - X_{i+1,j})^2 + (X_{i,j} - X_{i,j+1})^2}$$

$$dissimilarity = \|F - X\|_2^2$$

- 83 In the following subsections, the full procedures will be implemented.

##### 84 3.1.1 Calculate the partial differentiation of $\|X\|_{TV}$

- 85 The condition of the related blocks is shown in figure 5. According to the differential equation of the  
 86 equation of TV, for every  $x_{i,j}$ , there will be three part of equations connected with it, and they are

$$\begin{aligned} & \sqrt{(X_{i,j} - X_{i+1,j})^2 + (X_{i,j} - X_{i,j+1})^2}, \\ & \sqrt{(X_{i-1,j} - X_{i,j})^2 + (X_{i-1,j} - X_{i-1,j+1})^2}, \\ & \sqrt{(X_{i,j-1} - X_{i+1,j-1})^2 + (X_{i,j-1} - X_{i,j})^2}, \end{aligned}$$

87 and as you can see, there will be three inverse “L” shape block hit the center block for every  $x_{i,j}$ , and  
 88 each of them is colored with blue, green, and red.

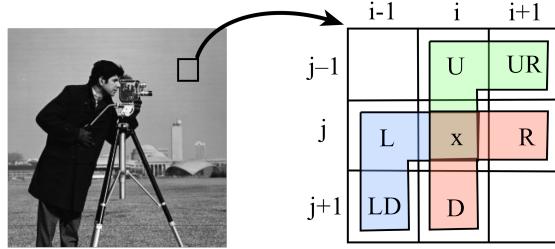


Figure 5: the representation of one center pixel surrounded by eight pixels.

89 By looking at the three equations connected with  $x_{i,j}$ , it can be found that they can be separated  
 90 to to part,  $\sqrt{(x-a)^2 + (x-b)^2}$  and  $\sqrt{(a-x)^2 + (a-b)^2}$ . To Calculate  $\frac{\partial}{\partial x} \|X\|_{TV}$ , the partial  
 91 diffrenciation of these two equation should be calculated first.

$$\frac{\partial}{\partial x}(\sqrt{(x-a)^2 + (x-b)^2}) = \frac{-a-b+2x}{\sqrt{a^2 - 2ax + b^2 - 2bx + 2x^2}}$$

$$\frac{\partial}{\partial x}(\sqrt{(a-x)^2 + (a-b)^2}) = -\frac{a-x}{\sqrt{(a-b)^2 + (a-x)^2}}$$

92 The Python implecation of these equations are shown below:

```

93
94 def delTV(L, R, U, D, LD, UR, x):
95     def delTV1(a,b,x):
96         divider = a*a - 2*a*x + b*b - 2*b*x + 2*x*x
97         return (-a-b+2*x)/math.sqrt(divider) if divider>0 else 0
98     def delTV2(a,b,x):
99         divider = math.sqrt((a-b)*(a-b)+(a-x)*(a-x))
100        return (-a+x)/divider if divider>0 else 0
101    return delTV1(D, R, x) + delTV2(L,LD,x) + delTV2(U,UR,x)
  
```

### 103 3.1.2 Calculate the partial differentiation of $\|F - X\|_2^2$

104 For this part, it can be solved easily.

$$\frac{\partial}{\partial x} ((a-x)^2) = 2x - 2a$$

## 105 3.2 Deal with the boundary condition

106 After implementing all the cells inside the image, there comes a problem that when meeting with the  
 107 boundary, all these equations will lose some information as shown in figure 6. For the shadow part,  
 108 they are the missing message.

109 It's not easy to deal with this problem, and for the current state, ignoring the boundary pixels is one  
 110 choice. This problem will be solved in future updates.

## 111 3.3 Implement the Gradient Descent for Total Variation Denoising

112 In the folder of code, *GD-general.ipynb* and *GD-general.py* can be found (the content of them is  
 113 the same). In the inner folder img, some original images can be seen, and they are downloaded  
 114 from <https://www.math.ust.hk/masyleung/Teaching/CAS/MATLAB/image/target2.html>. After running  
 115 *GD-general.ipynb*, the output files are saved in the inner folder *gen-img*, and the example of the  
 116 denoising processing are shown in figure 7. The Noised images are generated with Gauss Noise (GN).  
 117 The algorithm of Gauss Noise is in *img.py*.

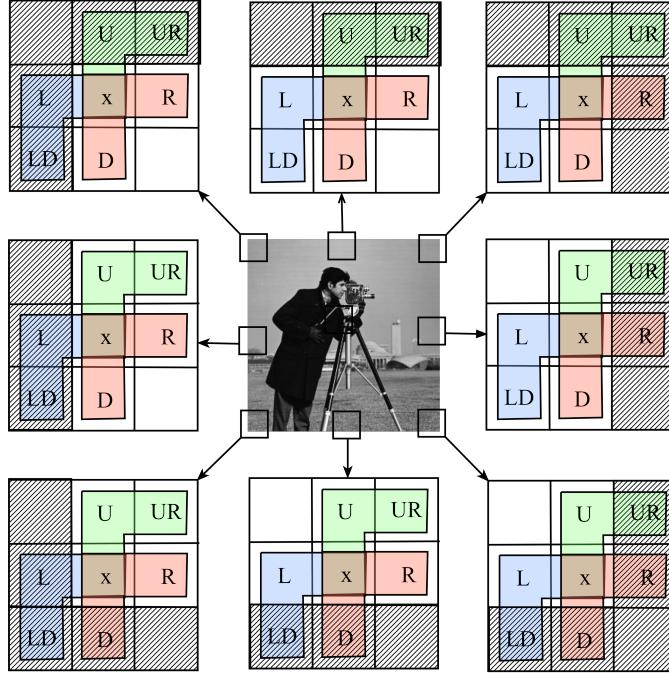


Figure 6: The boundary condition for the sample image which is hard to solve.

```

118
119     def gaussNoise(im_array, sigma):
120         im_array_flat = im_array.flatten()
121         for i in range(im_array.shape[0]*im_array.shape[1]):
122             pointInFlat = int(im_array_flat[i])+ random.gauss(0,sigma)
123             if pointInFlat < 0:
124                 pointInFlat = 0
125             if pointInFlat > 255:
126                 pointInFlat = 255
127             im_array_flat[i] = pointInFlat
128         im_array = im_array_flat.reshape([im_array.shape[0],im_array.
129                                         shape[1]])
130
131     return im_array

```

132 As shown in figure 7, the leftmost one is the original image directly downloaded from the website,  
133 and the next one is the noised image. Then the following three images are the denoised image with  
134 iteration 1, 10, and 100. It can be found that after 10 iterations, the noised image is already much  
135 better than the first iteration.

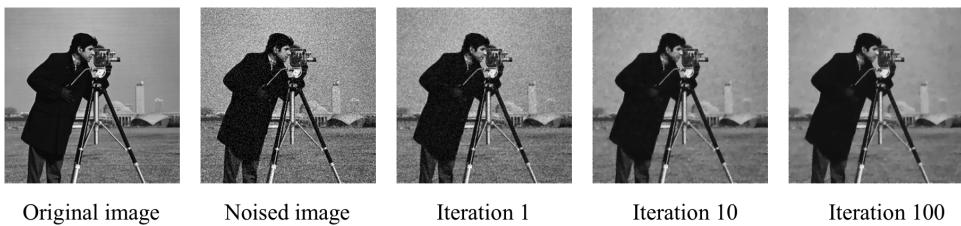


Figure 7: The original image, noised image, and three images are the denoised image with iteration 1, 10, and 100 with gradient descent.

136 To show the speed of convergence,  $f(x^k)$  vs. time and  $f(x^k)$  vs. iteration k are shown in figure 8  
137 and figure 9. It can be found that at first iterations, the speed of convergence is fast. However, after  
138 some iterations, the speed is much slower.

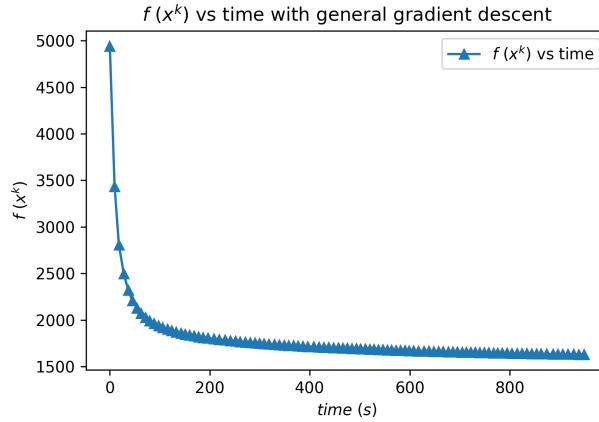


Figure 8:  $f(x^k)$  vs. time for general gradient descent.

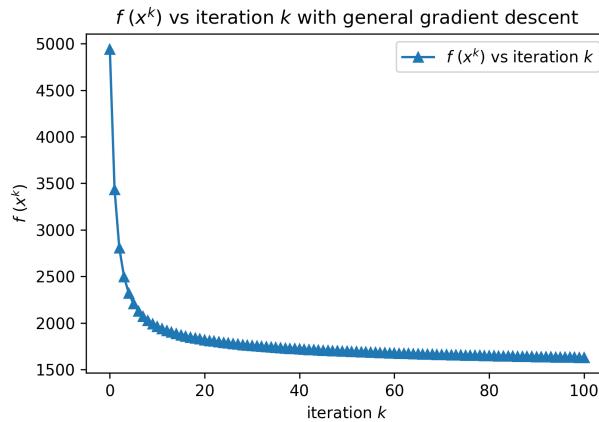


Figure 9:  $f(x^k)$  vs. iteration k for general gradient descent.

## 139 4 Optimization of the TV algorithm

### 140 4.1 Implement the Nesterov Acceleration to Accelerate Gradient Descent

141 To accelerate the speed of convergence, Nesterov acceleration is chosen as the speed-up algorithm.  
 142 And the thinking of methods is shown below.

$$\begin{aligned} z_{k+1} &= x_k - t_k \nabla f(x_k) \\ x_{k+1} &= z_{k+1} + \delta_k (z_{k+1} - z_k) \quad \delta_k \in [0, 1] \end{aligned}$$

143 And the algorithm is shown below [Ioannis, 2018].

144 Ioannis [2018] also Compared Polyak's method with Nesterov's algorithm. Ioannis says that Polyak's  
 145 method evaluates the gradient before adding momentum, while Nesterov's algorithm evaluates the  
 146 gradient after applying momentum as shown in figure 10 [Ioannis, 2018].

147 Just applying this method, the output is very similar to the pure exact gradient search (figure 11). But  
 148 the difference is not easy to be seen by human eyes. So the  $f(x^k)$  vs. time and  $f(x^k)$  vs. iteration k  
 149 are presented to show the speed of convergence, in figure 12 and figure 13. It can be found that at first  
 150 iterations, the speed of convergence is fast. However, after some iterations, the speed is much slower.

---

**Algorithm 2** Gradient Descent Algorithm

---

**procedure** GRADIENT DESCENT ALGORITHM WITH NESTEROV ACCELERATION

given a starting point  $x \in \text{dom } f$ , learning rate  $\delta_k$

repeat:

$$1. z_{k+1} = x_k - t_k \nabla f(x_k)$$

$$2. x_{k+1} = z_{k+1} + \delta_k (z_{k+1} - z_k) \quad \delta_k \in [0, 1)$$

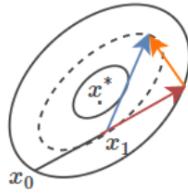
3. Line search. Choose step size  $t$  via exact or backtracking line search.

$$4. \text{Update. } x := x_{k+1}(t_k)$$

until stopping criterion is satisfied.

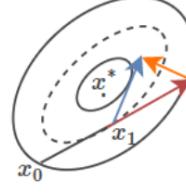
---

*Polyak's Momentum*



$$x_{t+1} = x_t - \alpha \nabla f(x_t) + \mu(x_t - x_{t-1})$$

*Nesterov Momentum*



$$x_{t+1} = x_t + \mu(x_t - x_{t-1}) - \gamma \nabla f(x_t + \mu(x_t - x_{t-1}))$$

Figure 10: Comparison of Polyak's method and Nesterov's algorithm.

151 **4.2 Comparison of gradient descent and gradient descent accelerated by Nesterov method**

152 Still, comparing those graphs in one figure is a better way to show the difference. Figure 14 shows  
153 the comparison of  $f(x^k)$  vs. time for gradient descent and gradient descent accelerated by Nesterov  
154 method.

155 When accelerated by the Nesterov method, it only cost 82 seconds to reach the value that general  
156 gradient descent needs to cost 918 seconds to reach, which is 10 times faster!

157 Comparing  $f(x^k)$  vs. iteration for gradient descent and gradient descent accelerated by Nesterov  
158 method as figure 15 present the same output: Nesterov method only cost 9 iterations to reach the  
159 value that general gradient descent need to cost 100 iterations to reach, which is also about 10 times  
160 faster!

161 **4.3 Implement the Stochastic Method to Accelerate Gradient Descent**

162 Because there are many pixels in one single image, it would be difficult to calculate the best descent  
163 direction, and for one direction, the step size would not be the best length for every pixel.

164 To make the selection of step size more efficient, it could be better to take apart the single image to  
165 multiple small images, and then do gradient descent for every small image separately.

166 A simple solution is to split an image to multiple  $n$  by  $n$  rectangles, and according to figure 5, there  
167 should be a margin as a helper to calculate the differentiation of the center split images.

168 To make the process simpler and easy to understand, we define one iteration as one full polling of all  
169 split blocks. And there will always process all blocks one by one with no multiprocess, which means  
170 in this case, the time would be the real time that needed to finish all the calculations without any trick.

171 First, let me to introduce you the  $f(x^k)$  vs. time and  $f(x^k)$  vs. iteration  $k$  are presented to show the  
172 speed of convergence, in figure 16 and figure 17, compared with the general gradient discent and the  
173 gradient discent accelerated by Nesterov method.

174 As you can see in figure 16, the overall time of 100 iterations for Gradient descent with the stochastic  
175 method is longer than the previous two methods, this is because there need to calculate the best step  
176 size with exact line search for  $(k/n)^2$  times, where  $k$  is the length of the image, and  $n$  is the size of

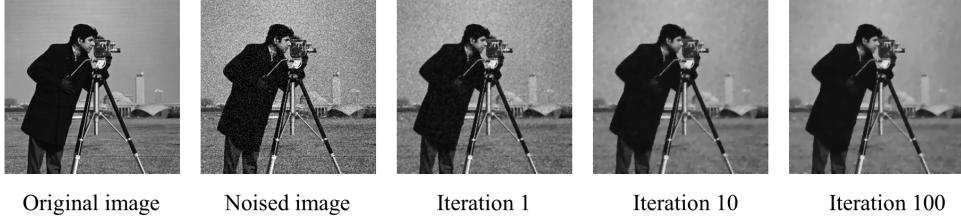


Figure 11: The original image, noised image, and three images are the denoised image with iteration 1, 10, and 100 with gradient descent accelerated by the Nesterov method.

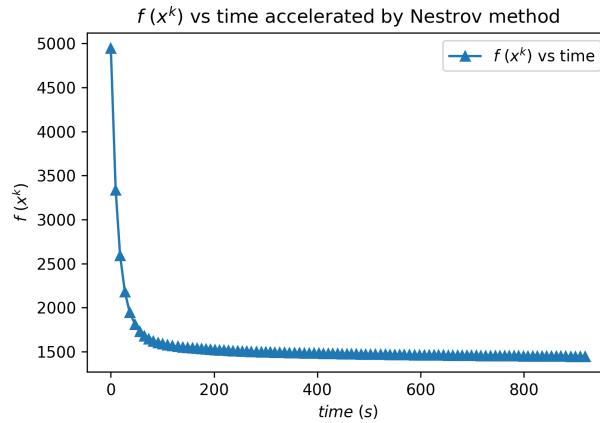


Figure 12:  $f(x^k)$  vs. time for gradient descent accelerated by Nesterov method.

177 the block you choose. Though the calculation of the best step size is much more than the previous  
 178 two solutions, it works better when time is long enough. This is because when some block with a low  
 179 score has already reached its best state, there can be many other blocks that can be improved largely  
 180 without being affected by those already converged blocks.

181 Considering figure 16, we can get the same conclusion, but due to the inconsistency of the time and  
 182 iteration of the stochastic method, it's better to consider only the  $f(x^k)$  vs. time figure.

183 Second, How will the block size  $n$  affect the speed of convergence? Here the size  $n$  means the block  
 184 is  $n \times n$  block. The block size 2, 4, 8, 16 are considered in the experiment. let me to introduce you  
 185 the  $f(x^k)$  vs. time and  $f(x^k)$  vs. iteration k are presented to show the speed of convergence, in figure  
 18 and figure 19,

187 Considering the  $f(x^k)$  vs. iteration k figure, it can be found that the speed of convergence increased  
 188 as the size of block decreased. This is because the smaller the block size, the easier the best step size  
 189 can reach its minimal value, and then the faster the image to fit its denoised state. However, looking  
 190 at figure 18, it will be found that the overall time for 100 iterations is very different for those block  
 191 sizes.

192 According to figure 18, the speed of convergence is fastest for block size 2, but the overall time for  
 193 block size 2 is the longest, which is almost the double of block size 4, or three times longer than the  
 194 block size 8. The block size 4 will cause the shortest overall time, and enlarge or reduce the size of  
 195 the block will both cause the overall time to be longer.

196 For all the previous experiments, the number of  $\lambda$  is 0.9. This is because for gradient descent and  
 197 gradient descent accelerated by the Nesterov method are both slow, which means for small  $\lambda$  like 0.1,  
 198 0.2, 0.5, the noised image cannot converge as fast as the user wants. But for the stochastic method,  
 199 the speed of convergence is fast, which causes the noised image being over-denoised, as shown in  
 200 figure 20.

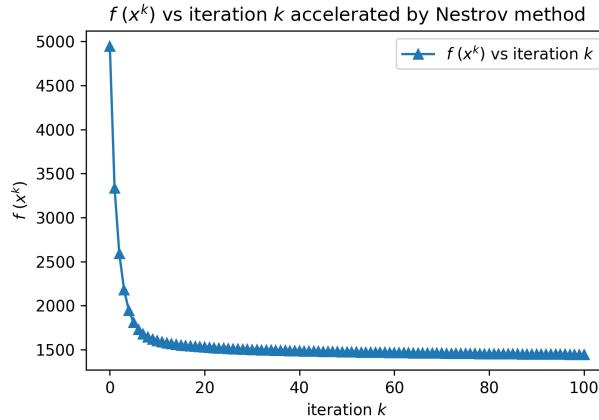


Figure 13:  $f(x^k)$  vs. iteration  $k$  for gradient descent accelerated by Nesterov method.

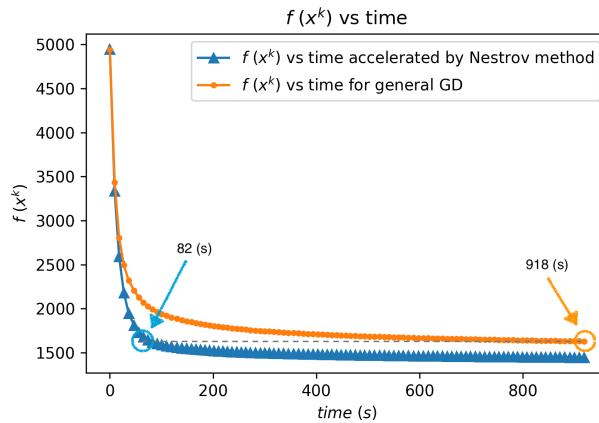


Figure 14: Comparison of  $f(x^k)$  vs. time for gradient descent and gradient descent accelerated by Nesterov method.

201 As you can see, as the decrease of block size, the noised image being smoother faster, but for block size  
 202 8 and 4, they are over-denoised. In this case, what we need to do is to decrease  $\lambda$ , and by experiment,  
 203  $\lambda = 0.1$  works well for block size 8 and 4.

## 204 5 Conclusion

## 205 References

- 206 Amir Beck and Marc Teboulle. Fast gradient-based algorithms for constrained total variation image  
 207 denoising and deblurring problems. *IEEE transactions on image processing*, 18(11):2419–2434,  
 208 2009.
- 209 Peter Blomgren, Tony F Chan, Pep Mulet, and Chak-Kuen Wong. Total variation image restora-  
 210 tion: numerical methods and extensions. In *Proceedings of International Conference on Image  
 211 Processing*, volume 3, pages 384–387. IEEE, 1997.
- 212 Dali Chen, YangQuan Chen, and Dingyu Xue. Fractional-order total variation image denoising based  
 213 on proximity algorithm. *Applied Mathematics and Computation*, 257:537–545, 2015.
- 214 Qiang Chen, Philippe Montesinos, Quan Sen Sun, and Peng Ann Heng. Adaptive total variation  
 215 denoising based on difference curvature. *Image and vision computing*, 28(3):298–306, 2010.

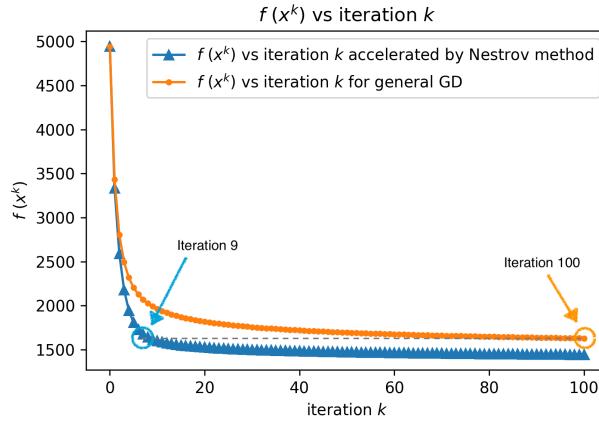


Figure 15: Comparision of  $f(x^k)$  vs. iteration k for gradient descent and gradient descent accelerated by Nesterov method.

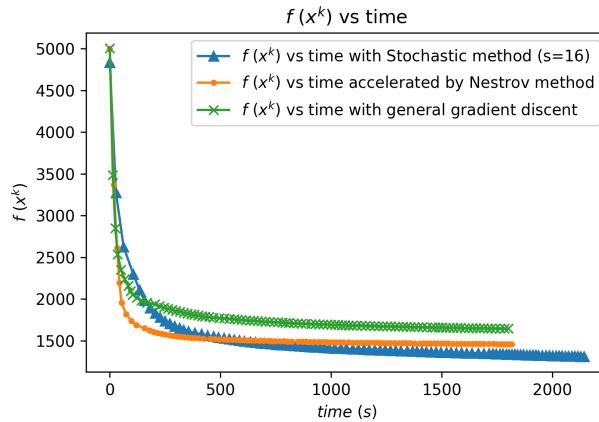


Figure 16: Comparision of  $f(x^k)$  vs. time for gradient descent, gradient descent accelerated by the Nesterov method, and gradient descent accelerated by the Stochastic method.

- 216 Laurent Condat. A direct algorithm for 1-d total variation denoising. *IEEE Signal Processing Letters*,  
217 20(11):1054–1057, 2013.
- 218 Sylvain Durand and Jacques Froment. Reconstruction of wavelet coefficients using total variation  
219 minimization. *SIAM Journal on Scientific computing*, 24(5):1754–1767, 2003.
- 220 Glenn R Easley, Demetrio Labate, and Flavia Colonna. Shearlet-based total variation diffusion for  
221 denoising. *IEEE Transactions on Image processing*, 18(2):260–268, 2008.
- 222 Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1.  
223 Springer series in statistics New York, 2001.
- 224 Mitliagkas Ioannis. Nesterov’s accelerated gradient, stochastic gradient descent. *IFT 6085 - Lecture*  
225 6, 2018.
- 226 Ivan W Selesnick, Ankit Parekh, and Ilker Bayram. Convex 1-d total variation denoising with  
227 non-convex regularization. *IEEE Signal Processing Letters*, 22(2):141–144, 2014.
- 228 DNH Thanh and SD Dvoenko. A denoising of biomedical images. *The International Archives of*  
229 *Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(5):73, 2015.

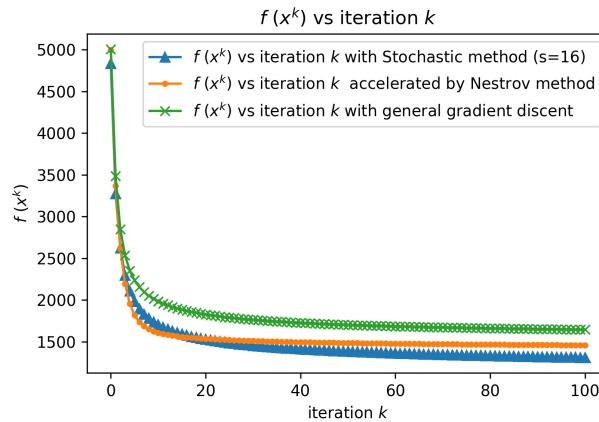


Figure 17: Comparision of  $f(x^k)$  vs. iteration  $k$  for gradient descent, gradient descent accelerated by the Nesterov method, and gradient descent accelerated by the Stochastic method.

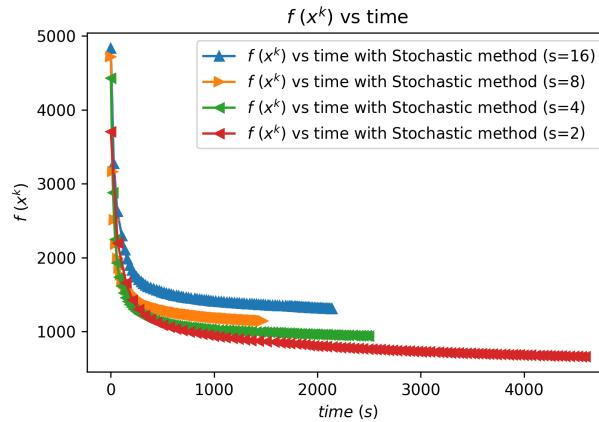


Figure 18: Comparision of  $f(x^k)$  vs. time for gradient descent accelerated by the Stochastic method with block size 2, 4, 8, 16.

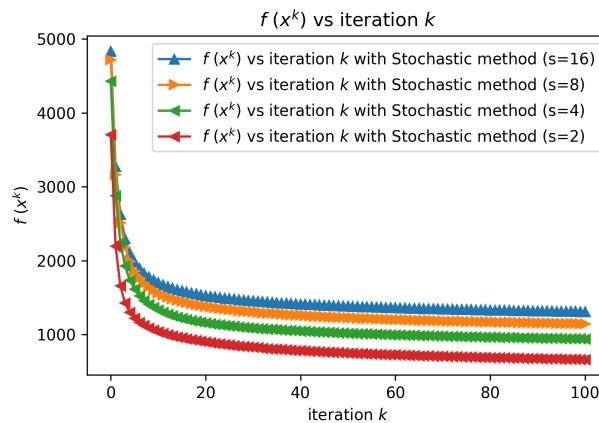


Figure 19: Comparision of  $f(x^k)$  vs. iteration  $k$  for gradient descent accelerated by the Stochastic method with block size 2, 4, 8, 16.

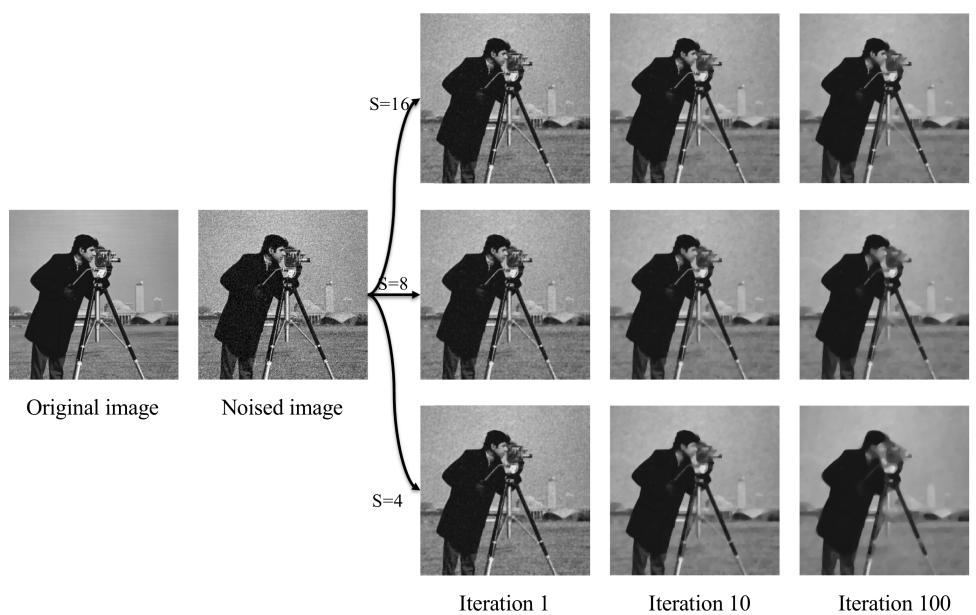


Figure 20: The original image, noised image, and three images are the denoised image with iteration 1, 10, and 100 with gradient descent accelerated by the Stochastic method with block size 4, 8, and 16.