Experiment.1: Image Upsampling

I. Task Introduction

We are given an image upsampling task. The details are as follows:

The original color image is represented as a 64×64×3 matrix of intensities I. The image is provided by "lena64color.tiff". The downsampled color image is represented as a 32×32×3 matrix of intensities D. The image is provided by "lena32color.tiff". Our job is to upsample the image D to be a 64×64 color image, by guessing the missing pixels. Generally, for a upsampling application, the original high-resolution image is not available. If the upsampled image will be represented by U, we can try to guarantee that U satisfies the condition:

$$U_{ij} = D_{mk},\ m = i/2+1,\ k = j/2+1.$$

1.1 Task specifications:

1. Please design the convex optimization model to derive the upsampled image. You can think about minimizing ℓ_1, ℓ_2 and ℓ_∞ norm and compare the effect of each of them;
2. Please try to introduce some regularization items to your model for some smooth regions to guarantee the smoothness in the variation of intensities, such as ℓ_2 regularization:

$$\sum_{i=2}^{64}\sum_{i=2}^{64}\left(\left(U_{ij}-U_{i-1,j}\right)^2+\left(U_{ij}-U_{i,j-1}\right)^2\right),$$

and ℓ_1 regularization:

$$\sum_{i=2}^{64}\sum_{i=2}^{64}\left(\left|U_{i,j}-U_{i-1,j}\right|+\left|U_{i,j}-U_{i,j-1}\right|\right).$$

3. Image I. cannot be used in the model.
4. Compare the upsampled image U with I. Use PSNR (Average PSNR of RGB).
5. Please provide:

(a) The .m file (including the code of calculating the PSNR performance) and the upsampled images;

(b) The report that describes your model, your implementation, results and performance comparison in detail.

II. Task Implementation

2.1 Basic Idea of Implementation

From the task specification we can derive the basic thought of this upsampling task:

1) Make sure all the pixels in D are distributed in U based on certain criterion;

2) Implement interpolation in other pixels apart from those mentioned in 1);

3) The interpolation must fulfill certain criterion as mentioned in 1.1.

Now, we take a look of the criteria once again:

ℓ_2 regularization: $\sum_{i=2}^{64}\sum_{i=2}^{64}\left(\left(U_{ij}-U_{i-1,j}\right)^2+\left(U_{ij}-U_{i,j-1}\right)^2\right)$,

ℓ_1 regularization: $\sum_{i=2}^{64}\sum_{i=2}^{64}\left(\left|U_{i,j}-U_{i-1,j}\right|+\left|U_{i,j}-U_{i,j-1}\right|\right)$.

Both are used to make sure that the upsampled picture is smooth enough by minimize the difference between neighboring pixels. Note that the ℓ_2 regularization is actually the **Frobenius norm** of the difference between pixels.

2.2 Image Preprocessing

First, we use imshow() to check the given 64*64 image I, 32*32 image D and pre-extended image U as shown below.

Fig.1 Image Preprocessing

From Fig.1 we can easily see how pixels of D are distributed in U. And our mission is to smooth the interpolation in those black positions of U.

2.3 Convex Optimization Model

2.3.1 Basic models

In this section, we will derive several convex optimization models to fulfill this image upsampling task. **Note that all the objective functions in this section are sum of norms, i.e. convex functions.**

First, we decide to simply use the criteria mentioned before as our objective function. The convex optimization model using Frobenius norm regularization is given as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=2}^{64}\sum_{i=2}^{64}\left(\left(U_{ij}-U_{i-1,j}\right)^2+\left(U_{ij}-U_{i,j-1}\right)^2\right) \\ \text{subject to} \quad & U_{ij}\geq 0 \\ & U_{ij}\leq 255 \\ & U_{i-1.j-1}=D_{i/2,j/2},\ i=2,3,\ldots,64,\ \ j=2,3,\ldots,64 \end{aligned} \tag{1}$$

or using ℓ_1 norm regularization:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=2}^{64}\sum_{i=2}^{64}\left(\left|U_{i,j}-U_{i-1,j}\right|+\left|U_{i,j}-U_{i,j-1}\right|\right) \\ \text{subject to} \quad & U_{ij}\geq 0 \\ & U_{ij}\leq 255 \\ & U_{i-1.j-1}=D_{i/2,j/2},\ i=2,3,\ldots,64,\ \ j=2,3,\ldots,64 \end{aligned} \tag{2}$$

and ℓ_∞ norm regularization is also considered as:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=2}^{64}\sum_{i=2}^{64}\max\left(\left|U_{i,j}-U_{i-1,j}\right|+\left|U_{i,j}-U_{i,j-1}\right|\right) \\ \text{subject to} \quad & U_{ij}\geq 0 \\ & U_{ij}\leq 255 \\ & U_{i-1.j-1}=D_{i/2,j/2},\ i=2,3,\ldots,64,\ \ j=2,3,\ldots,64 \end{aligned} \tag{3}$$

Note that we **do not use** ℓ_2 **norm** because it requires calculation of eigenvalues, which would slow down the optimization process. Hence, we decide to abandon it hereafter in this experiment.

2.3.2 Objective function Melioration

However, we **cannot** directly optimize these three models derived in 2.3.1 with cvx toolbox because the calculation of the objective function is far too laborious and **time consuming**. That is to say, we have to change the form of the objective functions with **matrix form**. During optimization process, the above objective functions are, in fact, **norms of the difference matrices** as shown below.

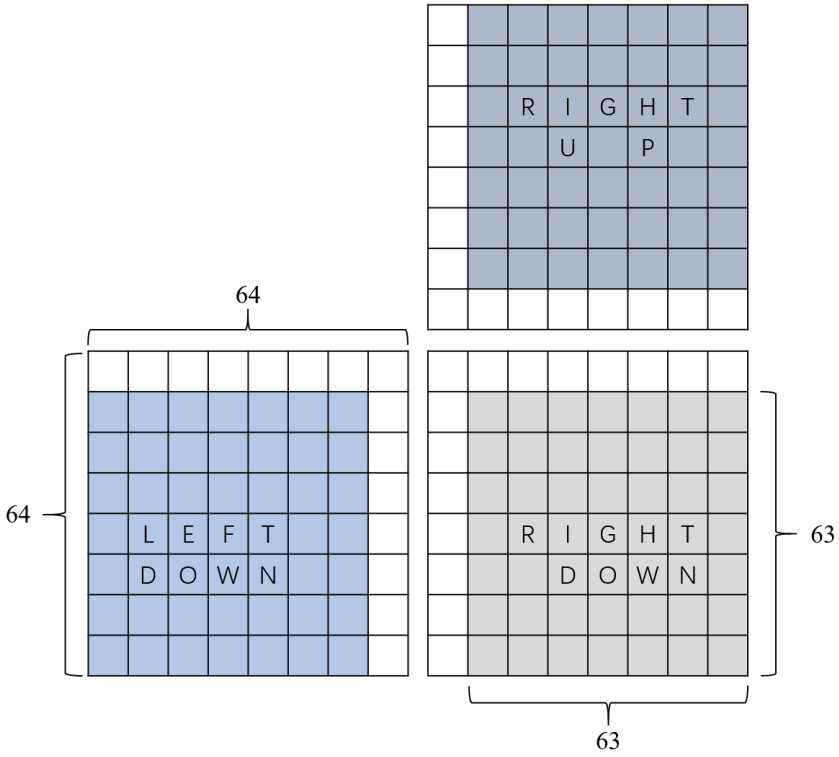

Fig.2 Illustration of the objective function

We denote the difference of RIGHT-DOWN matrix minus LEFT-DOWN matrix as $diff_d$ and the difference matrix of RIGHT-DOWN matrix minus RIGHT-UP matrix as $diff_c$. Hence the convex optimization model becomes:

$$
\begin{aligned}
&\text{minimize} \quad \|diff_c\|_1 + \|diff_d\|_1 \\
&\text{subject to} \quad U_{ij} \geq 0 \\
&\qquad\qquad U_{ij} \leq 255 \\
&\qquad\qquad diff_c = U_{2:64,2:64} - U_{1:63,2:64} \\
&\qquad\qquad diff_d = U_{2:64,2:64} - U_{2:64,1:63} \\
&\qquad\qquad U_{i-1.j-1} = D_{i/2,j/2}, \ i = 2,3,\ldots,64, \ \ j = 2,3,\ldots,64
\end{aligned}
\tag{4}
$$

and the other two objective function are $\|diff_c\|_\infty + \|diff_d\|_\infty$ and $\|diff_c\|_1 + \|diff_d\|_1$.

2.3.3 Further improvement

Improvement I:

It is naturally to think of changing the weights of $\|diff_c\|_p$ and $\|diff_d\|_p$, where $p = 1, 2, \ldots, \infty$. Therefore, the reformed convex optimization model can be stated as:

$$\begin{aligned} &\text{minimize} \quad \alpha\|diff_c\|_p + (1-\alpha)\|diff_d\|_p \\ &\text{subject to} \quad \alpha \geq 0,\ \alpha \leq 1 \\ &\qquad\qquad U_{ij} \geq 0,\ U_{ij} \leq 255 \\ &\qquad\qquad diff_c = U_{2:64,2:64} - U_{1:63,2:64} \\ &\qquad\qquad diff_d = U_{2:64,2:64} - U_{2:64,1:63} \\ &\qquad\qquad U_{i-1.j-1} = D_{i/2,j/2},\ i = 2,3,\ldots,64,\ j = 2,3,\ldots,64 \end{aligned} \tag{5}$$

Improvement II:

The main idea of using the above objective function is to assure the **smoothness** the image. But does the difference in these two directions can satisfy our demand on smoothness? Obviously, it is also very natural to add some difference terms or difference matrices in other directions.

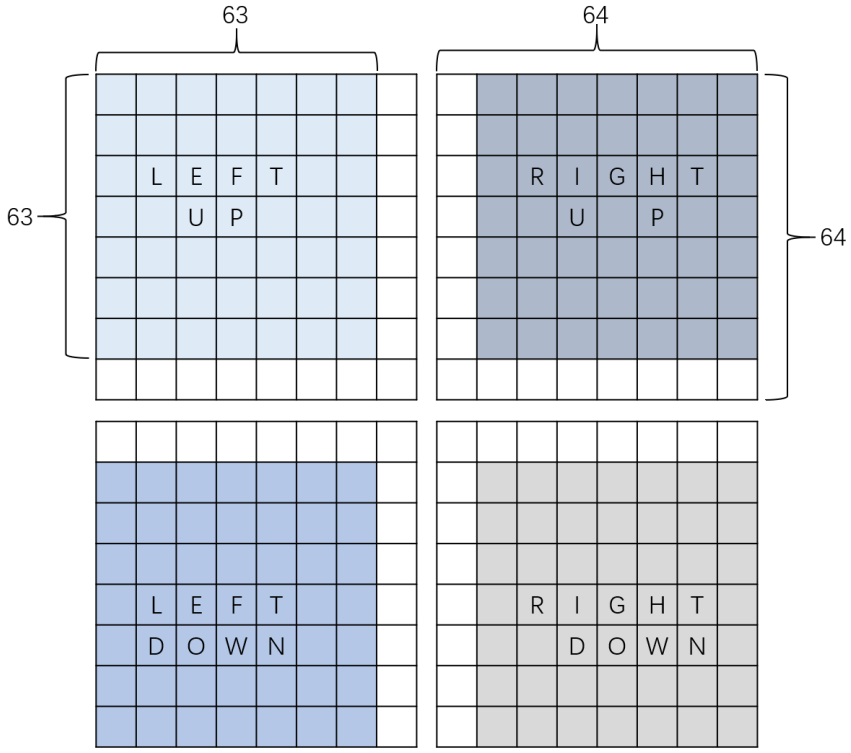

Fig.3 Difference in four directions

We have taken difference matrices in two directions in above models. Likewise, we take difference matrices of another directions. Now, we have difference matrices in four directions: LEFT-UP minus RIGHT-UP, LEFT-UP minus LEFT-DOWN, RIGHT-DOWN minus RIGHT-UP and RIGHT-DOWN minus LEFT-DOWN, denoted as $diff_a$, $diff_b$, $diff_c$, $diff_d$ respectively. We can also add weights of these four difference matrices. Hence the final convex optimization model is expressed as:

$$\begin{aligned} \text{minimize} \quad & \alpha\|diff_a\|_p + \beta\|diff_b\|_p + \gamma\|diff_c\|_p + \theta\|diff_d\|_p \\ \text{subject to} \quad & \alpha \geq 0,\ \beta \geq 0,\ \gamma \geq 0,\ \theta \geq 0 \\ & U_{ij} \geq 0,\ U_{ij} \leq 255 \\ & diff_a = U_{1:63,1:63} - U_{1:63,2:64} \\ & diff_b = U_{1:63,1:63} - U_{2:64,1:63} \\ & diff_c = U_{2:64,2:64} - U_{1:63,2:64} \\ & diff_d = U_{2:64,2:64} - U_{2:64,1:63} \\ & U_{i-1.j-1} = D_{i/2,j/2},\ i = 2,3,\ldots,64,\ \ j = 2,3,\ldots,64 \end{aligned} \tag{6}$$

We also implement the open operation to the optimized image so that the output image could show a little bit better performance.

2.4 Results and Evaluation

In section 2.3 we presented 3(or 4) models, in which the objective functions can be either ℓ_1, ℓ_∞ or Frobenius norm. That is to say, we have 9 (or 12) models for this image upsampling task. Here we use cvx toolbox developed by Stephen Boyd within MATLAB and we use imresize(D, 2, 'bilinear') as a **benchmark**. The performance of these models is evaluated by the PSNR value of the optimized image U. The results are shown as follows.

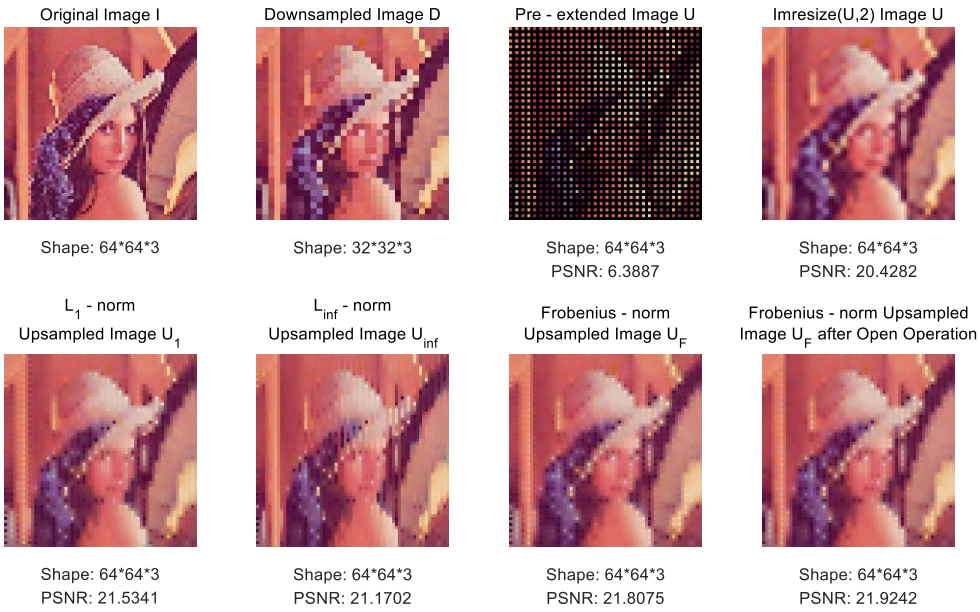

Fig.4 basic model (i.e. equation (4))

From Fig.4 we can see that using Frobenius norm of difference matrices as objective function can have the best performance among other norms with PSNR = 21.8075 and 21.9242 after open operation while the default bilinear interpolation gives an image with PSNR = 20.1282.

Then we show the image generated by improved model I with the optimal $\alpha = 0.614.$ The Frobenius objective function still outperforms among other norm objective functions, with PSNR = 21.8518 and 21.9397 after open operation.

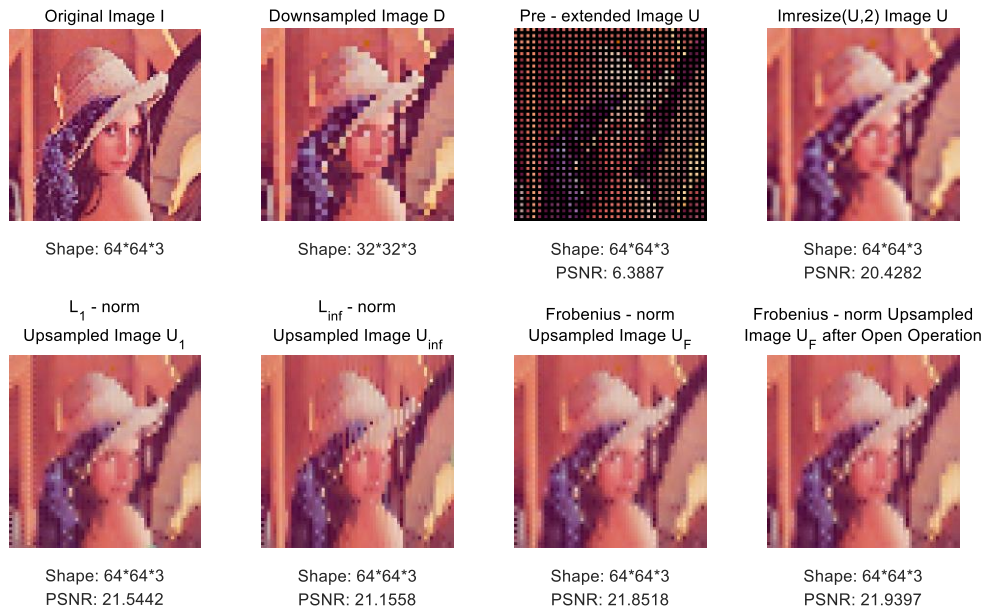

Fig.5 Improved model I (optimal $\alpha = 0.614.$)

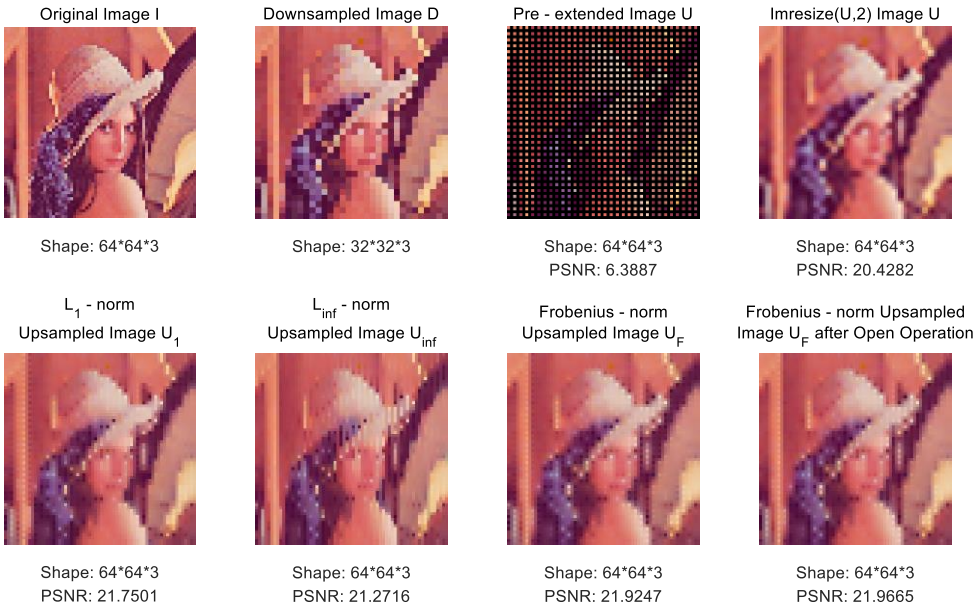

Fig.6 Improved model II ($\alpha=\beta=\gamma=\theta=1$)

Next, we show the image generated by improved model II with four equal weights in Fig.6 and the result just proves our assumption in 2.3 that using difference matrices of four directions is better than using matrices of just two directions. Then we can find the optimal value of the four weights by dichotomy and the optimal values of these four weights are about $\alpha=1,\ \beta=2.325,\ \gamma=1,\ \theta=1$. The generated images by model with these weights are shown below.

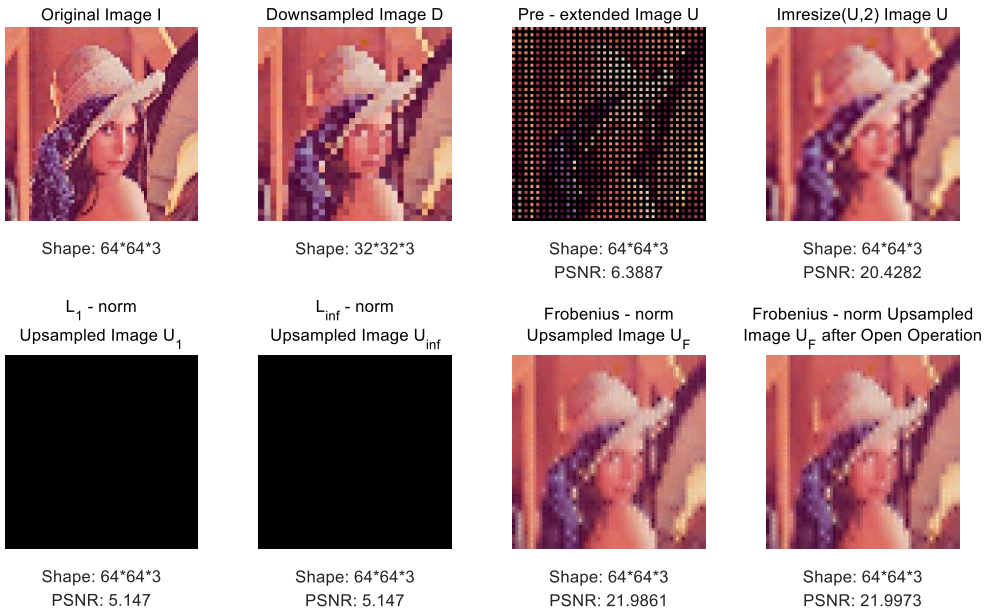

Fig.7 Improved model II (optimal $\alpha=1,\ \beta=2.325,\ \gamma=1,\ \theta=1$)

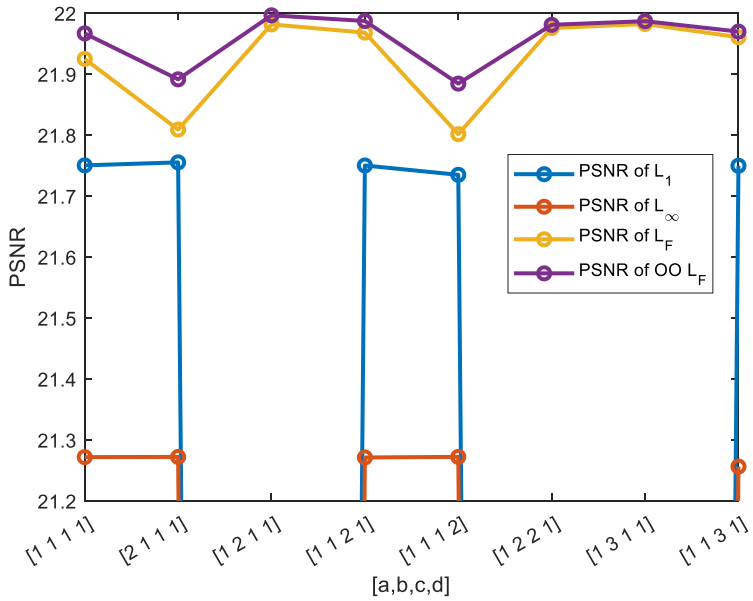

Fig.8 Changes of PSNR with different weights

The other parameter changes and corresponding PSNR values are shown in Fig.8. We can easily find that **the model of Frobenius norm always shows the best performance, even in cases where models of other two norms cannot converge.** Therefore, in the procedure of finding optimal value of β, we just focus on model with Frobenius norm. The values of β and corresponding PSNR are shown in Fig.9 as below.

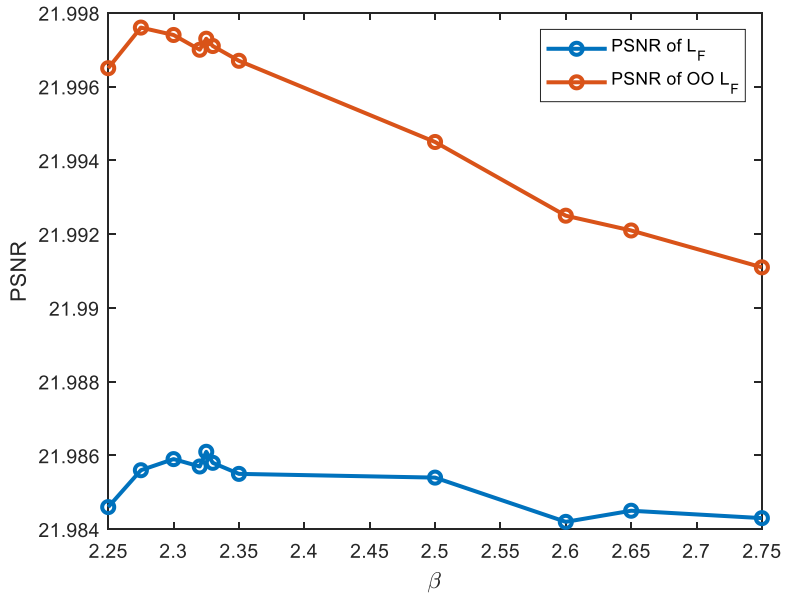

Fig.9 Finding the optimal β

From above modeling and experiments, we can give the optimal PSNR of different models as shown in Table.1. The overall best PSNR of upsampled image is generated by Frobenius norm model and the **overall optimal PSNR is 21.9973.**

Table. Best PSNR of different model

	Bilinear (Benchmark)	basic model	Improved model I (optimal α = 0.614)	Improved model II ($\alpha = \beta = \gamma = \theta = 1$)	Improved model II (optimal β = 2.325)
PSNR	20.4282	21.8075	21.8518	21.9247	21.9861
PSNR (after Open Operation)		21.9242	21.9397	21.9665	21.9973

III. Potential Improvement

Potential improvements are adding a factor of regularization of the image itself, or change parameters for different RGB channel, or adding second-order difference matrices as objective function, etc. During this experiment, I have tried some more objective functions but none showed better performance than the given results above. Due to limited time and energy, I cannot implement all potential improvements and amend them to the best performance. **Therefore, trials of these are not shown in this report but codes are still left in .m files.**

IV. Conclusions

In this experiment I developed 9 (or 12) convex optimization models for this image upsampling task. They are basic smoothing models and weights-changing models, improved smoothing models that consider smoothness in more directions and improved weights-changing models of course. As the models are reformed improved gradually, they show better performance on the PSNR of generated images with a best PSNR value of **21.9973**. This proves that the improvements are practical and useful. One interesting thing I found during this experiment is that the **model with Frobenius norm as objective function shows the best performance even in cases where models with other norm as objective function cannot converge.** This shows that Frobenius norm might be quite useful in some image processing tasks.

From this experiment I have deepened knowledge of convex optimization. First,

the form of the model is very crucial. At first, I implemented for-loop in the objective function, which results in very long running time. However, after changing the for-loop into norm of matrices, the running time clearly decreased a lot.

Then, the knowledge of image processing does help in the task of upsampling. Smoothing and Open operation (erosion and dilation) are the main idea of my models. I also tried filtering, sharpening and histogram equalization, etc. However, none of these techniques show remarkable improvement in PSNR value because the given image D has only 25% of the information in original image I. Therefore the traditional image processing techniques are bounded by this lack of information.

This sheds light on the significance of deep learning for us. With powerful deep neural networks like ResNet or DenseNet, we can absolutely do better than the current results. Though this might take much time and has certain requirements of data.

However, this does not mean that convex optimization has no benefits at all. In fact, in many circumstances convex optimization shows better performance than deep learning and the thoughts of convex optimization are in fact within many machine learning theories. Convex optimization is the very foundation of many theories and disciplines, it is worth spending more time to dig into it.