Image Upsampling via Modified Imposed Edge Statistics

Weixi Zhang, wz1219, N18406096

Zhimiao Lai, zl2083, N11140147

May 15, 2018

Abstract

In our project, we implemented an image-upsampling algorithm by Raanan Fattal, 2007 using MATLAB. To get the conditional distribution of the error images, we extract the feature vectors (d, s, m) from 10 indoor images. Instead of calculating the conditional distribution for any arbitrary configuration of (d, s, m), we perform quantization to the space of (d, s, m). This can enable us to use less images to get the conditional distributions. Thus, it requires less computation and time to train the statistical models. With the distributions we have, we can predict a new indoor image by adding the averaged error image to a bicubic-upsampled image. Our result shows that the upsampled image by our algorithm has a stronger impose on the edges than bicubic-upsampled images.

1 Background

Image resizing and interpolation is one of the most elementary image operation. And it is applied with many purposes in many areas. The applications range from 2D to 3D, from images to videos.

Upsampled images methods often face with several major challenges: blurry effect on sharp edges, lack of texture details, the ‘jaggies artifact, and the ringing effect. [2]

Blurring is a mismatch between an intermediate data representation and their final downsampled or oversampled version. In this case, the mismatch is such that the intermediate data is too coarse for the task. This results in an image that appears to be out of focus.

Blocking arises when the support of the interpolant is finite. In this case, the influence of any given pixel is limited to its surroundings, and it is sometimes possible to discern the boundary of this influence zone.

Ringing arises because most good synthesis functions are oscillating.

Our project is mostly based on Fattal’s paper [1] . We also employed several methods to fill in the implicit details of the algorithm and to make some improvement on computation. In the following section we discuss the previous work mentioned in Fattal’s paper and apply some of the ideas into our implementation. In the third section we revisit Fattal’s algorithm, focusing on the basic concept and procedure. In Section 4, we introduce our implementation based on Fattal’s basic idea. In Section 5, we analyze this algorithm and discuss our output performance.

2 Previous Work

Prior to Fattal, upsampling methods are often based on certain assumptions that images are smooth or band-limited. But for most images, these assumptions are highly inaccurate and yields bad performances on the output. [1]

The most basic and classic method are the linear filters, including Nearest-Neighbor, Bilinear, Bicubic, Hann, Hamming, and Lanczos interpolation kernels. They can be described as

$$\mathfrak{f}(x) = \sum_{k \in Z^q} f_k \varphi_{int}(x - k) \qquad \forall x = x_1, x_2, \dots \dots, x_q \in R^q$$

where an interpolated value $\mathfrak{f}(x)$ at some (perhaps non-integer) coordinate x in a space of dimension q is expressed as a linear combination of the samples $\mathfrak{f}_k$ evaluated at integer coordinates $k = k_1, k_2, \dots, k_q \in Z^q$, the weights being given by the values of the function $\varphi_{int}(x - k)$. A classical example of the synthesis function φ_{int} is the sinc function, in which case all synthesized functions are band-limited. The construction of these kernels relies strongly on the assumption that the image data is either spatially smooth or band-limited.[2]

In the later years, more sophisticated methods, where interpolation weights adapt locally to the image data, have been developed to reduce blurriness and other artifacts. [1]

$$\mathfrak{f}(x) = \sum_{k \in Z^q} c_k \varphi(x - k) \qquad \forall x \in R^q$$

This offers new possibilities, in the sense that interpolation can now be carried in two separate steps: firstly, the determination of coefficients c_k from the samples $\mathfrak{f}_k$, and secondly, the determination of desired values $\mathfrak{f}(x)$ from the coefficients c_k .Nevertheless, since the interpolation weights are estimated from flat regions as well, edges do not appear as sharp as they should. [2]

Another set of methods are implemented by storing additional image data.

[3] keeps infinitely sharp edges by storing discontinuity graphs, and if the input image has a planar nature, this method gives a good performance on recovering edges.

Some other methods reached the optimal solution on edges and blocking effects by iteration or by pyramidal image representation. Instead of assuming a smoothness prior for the underlying intensity function, [4] assumes smoothness of the level curves. These methods can do better job on removing blurring effects but suffers from ringing artifacts.

Also, if we have several images differs in camera angle, we can collect and merge their information in order to estimate the texture of an object which appears in more than one image. [5] did this after stitching different angle pictures to normalize and improve the resolution of a panorama image.

Image-based models for computer graphics lack resolution independence: they cannot be zoomed much beyond the pixel resolution they were sampled at without a degradation of quality. Interpolating images usually results in a blurring of edges and image details. [6] describe image interpolation algorithms which use a database of training images to create plausible high-frequency details in zoomed images. The use, however, of such a finite non-parametric set of examples tends to introduce some irregularities into the constructed image.

Another method based on the idea that an image can be decompose into two parts: cartoon and texture(or noise). [7] Then the total variation minimization can be applied on the texture part.

3 Fattal's Algorithm

Fattal's paper proposed an upsampling method that given a low-resolution image, if we can find every single pixel's closest edge, we can extract several features, including the distance to the edge, and the edge's scattering and the variance of intensity. This is based on the observation that a sharp edge in an image corresponds to relatively large intensity gradients concentrated along the edge, while a smooth edge is composed of a more scattered set of weaker gradients.

Another novel idea is they upsampled an image by constructing its gradient field rather than determining the pixel intensities directly. The sharp edges can be constructed while avoiding ringing effects in this way. To follow this approach, we are required to predict the spatial intensity differences at the high-resolution based on the low-resolution input image. To do this, Fattal adjusted the previous idea that which trained a set of images, Fattal do the training and prediction the intensity distribution in gradient scale.

Given an image I, we unsample it to get $\tilde{I}$. The spatial derivatives and the gradient norm are evaluated at every pixel $x = (x, y)$ as following

$$\tilde{I}_x(x) = \tilde{I}(x+1, y) - \tilde{I}(x-1, y),$$

$$\tilde{I}_y(x) = \tilde{I}(x, y+1) - \tilde{I}(x, y-1),$$

and $N(x) = \left|\left|\nabla\tilde{I}\right|\right|_2$, where $\nabla\tilde{I} = (\tilde{I}_x(x), \tilde{I}_y(x))$.

The ray in the gradient direction is parameterized at each pixel by

$$\varphi_x(t) = x + t \cdot \nabla\tilde{I}(x)/N(x).$$

This is used to find and measure the closest edge from x. To extract the edge features nearest to the pixel x, we should do exhaustive search along the gradient direction. Along the gradient direction φ_x, we first find the local maxima of N, which is the peak of the nearest edge, we define it as t_0. As we move away from t_0, the points t_1 and t_2 are then taken to be the points where φ_x ceases to decrease This search can be performed only within the limits of $|t| \leq 5$ (pixels).

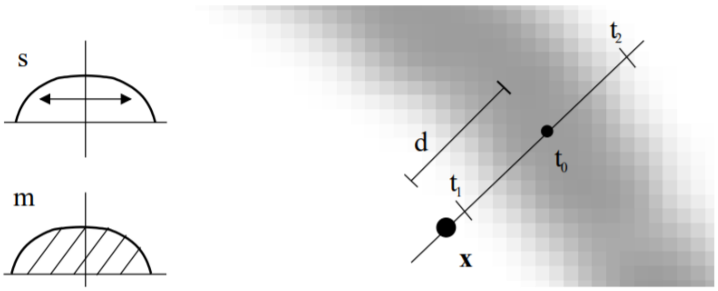

Figure 1: Illustration of d, m, s

Then the edge features extracted from the low-resolution image can be defined as follow:

1. The total change in intensity:

$$m(x) \ = \ \int_{t_1}^{t_2} N(\varphi_x(t)) \, dt.$$

2. The distance from the closest edge:

$$t_0 \ = \ \int_{t_1}^{t_2} N\big(\varphi_x(t)\big) \, dt / m(x) \, ,$$

$$\text{and } d(x) \ = \ |t_0|.$$

We also store

$$r(x) \ = \ sign(t_0)$$

which indicates on which side of x (along φ_x) the edge passes.

3. The closest edge's spatial scattering:

$$s(x) \ = \ \int_{t_1}^{t_2} (t - t_0)^2 \cdot N\big(\varphi_x(t)\big) \, dt / m(x).$$

We use discrete sums to approximate the integrals above.

We also define the local transformation matrix

$$F(\nabla \tilde{I}(x)) \ = \left\| \nabla \tilde{I}(x) \right\|^{-1} \begin{pmatrix} r(x) \cdot \tilde{I}_x(x) & -\tilde{I}_y(x) \\ r(x) \cdot \ \tilde{I}_y(x) & \tilde{I}_x(x) \end{pmatrix}.$$

The continuity measure between pixel x and its neighbor, $x + u$, is defined by

$$\ell(x, u) \ = \ \tilde{I}(x) - \tilde{I}(x + F(\nabla \tilde{I}(x)) \cdot u),$$

Under the assumption of invariance to translation and rotation, implying that there is no dependency on particular pixel position or edge orientation, we define the *edge-frame continuity modulus* (EFCM) as the conditional mean and variance of ℓ

$$\mu(u|[d, m, s]) = E[\ell(x, u)|[d, m, s]]$$

$$\sigma^2(u|[d,m,s]) = E[\ell(x,u) - \mu(u|[d,m,s])^2|[d,m,s]]$$

Above procedure can be applied to a set of images, and we'll get a database of μ, σ^2 of different u, d, m, s.

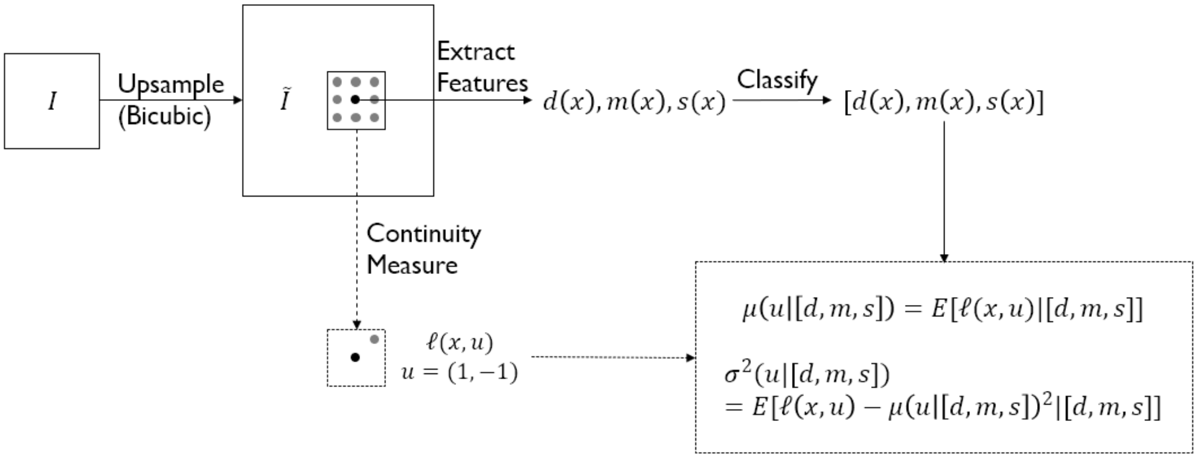

Figure 2: Training procedure

For a given offset u and parameters d, m and s, Fattal modeled the distribution of random variable ℓ by the Normal distribution

$$\ell(x,u) \sim \mathcal{N}(\mu(u|[d,m,s]), \sigma^2(u|[d,m,s]))$$

With the database of ℓ in terms of d, m, s, and u established, for a new input low-resolution image, we can find its best fit distribution, which is an optimization problem.

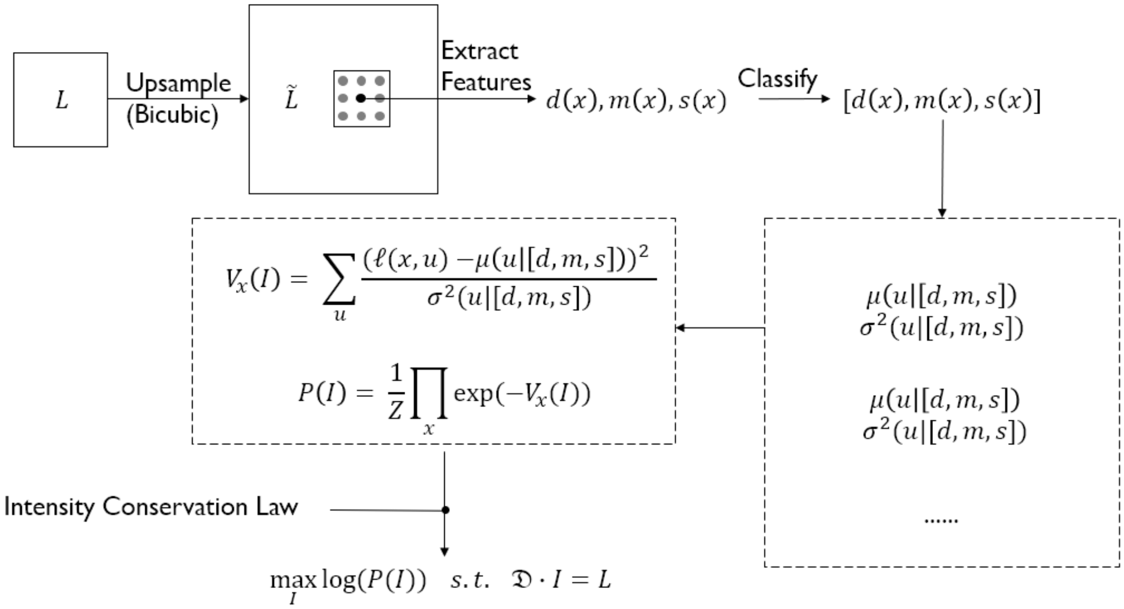
L
Upsample
(Bicubic)
$\tilde{L}$
Extract
Features
$d(x), m(x), s(x)$
Classify
$[d(x), m(x), s(x)]$
$V_x(I) = \sum_u \frac{(\ell(x,u) - \mu(u|[d,m,s]))^2}{\sigma^2(u|[d,m,s])}$
$P(I) = \frac{1}{Z}\prod_x \exp(-V_x(I))$
$\mu(u|[d,m,s])$
$\sigma^2(u|[d,m,s])$
$\mu(u|[d,m,s])$
$\sigma^2(u|[d,m,s])$
......
Intensity Conservation Law
$\max_I \log(P(I)) \quad s.t. \quad \mathfrak{D} \cdot I = L$

Figure 3: Testing procedure

This modeling of the EFCM gives rise to a Gauss-Markov Random Field model, and then we get the following Gibbs distribution

$$P(I) = \frac{1}{Z}\prod_{x}\exp(-V_x(I))$$

$$\text{with } V_x(I) = \sum_u \frac{(\ell(x,u) - \mu(u|[d,m,s]))^2}{\sigma^2(u|[d,m,s])}$$

The resulting distribution $P(I)$ predicts the conditional probability of I given the predicted edges behavior.

Another piece of information we can use is the intensity conservation law, that the result by downsampling I to the input image resolution must be identical to the input image L itself.

Thus, the upsampled image I is obtained as the solution of

$$\max_I \log(P(I)) \quad s.t. \quad \mathfrak{D} \cdot I = L$$

This form of constrained optimization problem can be solved using the method of Lagrange Multipliers.

The process above, as we can see, requires a large computation complexity. And if we want to upsample the image by more than factor of 2, this method must be called iteratively, i.e., for each time an input image can only be upsampled by 2 and then the output will be reapplied to the system as a new input.

The drawbacks of this method, as drawn from its testings, are the following: (i) the resulting sharp-edged images tend to emphasize lack of texture and absence of fine-details. (ii) the jaggies artifact, though minimal relative to the gain in sharpness, is sometimes noticeable in the presence of clear edges produced. (iii) acutely twisted edges, although scarce, are not captured by the EFCM and are undetected by conditioning on d, m, s alone. And (iv) the proposed method involves more computations than some of the existing techniques.

4 Our implementation

Code:

We implemented the algorithm in MATLAB.

10 indoor images (128x128, grayscale) were used to train the conditional distributions. Instead of estimating the conditional distribution of the pixel differences for any possible configuration of d,m,s, we tried to reduce the number of distributions by quantize the DMS space. By DMS space, it simply means the space of the features d, m, s. If we estimate the conditional distribution of the pixel differences for all possible configurations of d, m, s, in most cases, we only have only a small amount of pixel samples that have the same d, m, s, which means that the conditional distribution is calculated only on a small amount of data points!

To some extent, the problem here can be viewed as a clustering problem. Fattal's method is to view each configuration of d, m, s as one cluster. We tried to use two different methods: K-means and quantization, to cluster the d, m, s features.

Below, we visualize the d, m, s space and the result of K-means:

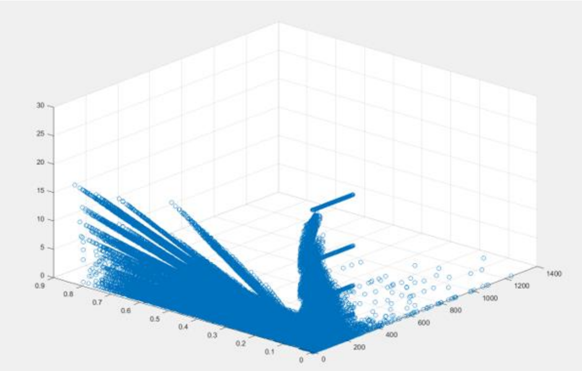

Figure 4: DMS space of 10 indoor images

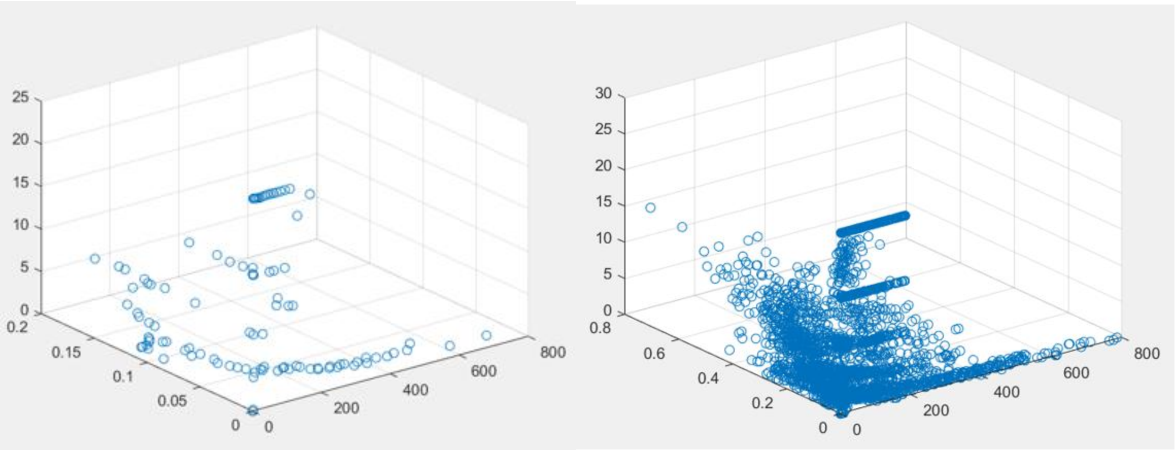

Figure 5: K-means with K = 100(Left) and K = 1000(Right)

Obviously, even we assume that there are 1000 clusters in the DMS space, it's still not enough to simulate the actual DMS space. So, we tried the second method: quantization the DMS space.

In our implementation, we divide the DMS space into 20-by-20-by-20 blocks. Based on the data points in a particular block, we can calculate the center of these points.

On the other hand, we defined the window size for the continuity measures as 3-by-3, which means that we have 8 u's: [1 1; 0 1; -1 1; -1 0; -1 -1; 0 -1; 1 -1; 1 0]. For each u, we can define its corresponding pixel difference image based on previous definition.

Recap: For the upsampled image, now we have one corresponding DMS space and 8 difference images.

Now we can estimate the conditional distribution of all possible DMS blocks (8000 in our implementation) for each difference image.

From what we've learned from the paper, we know that the conditional distribution of a particular choice of u, d, s, m can be modeled by a Gaussian distribution. To verify that quantization will not change this property, we visualize the distribution of the difference image of a particular choice of u and block (d, s, m).

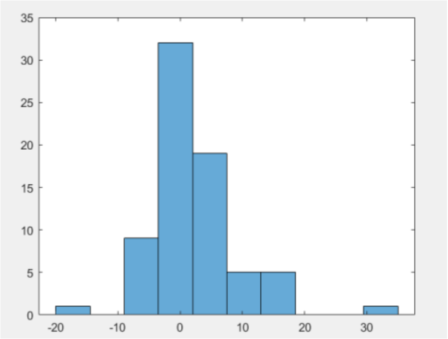

Figure 6: the distribution of pixel difference of u = [1,1] and d, s, m in the block

of d: from 39.5101 to 79.0201, s: from 0.0377 to 0.0754, m: from 1.2500 to 2.5000

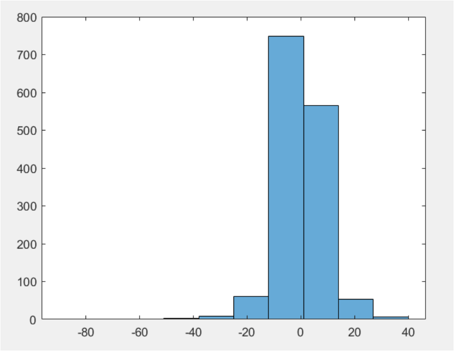

Figure 7: the distribution of pixel difference of u = [1,1] and d, s, m in the block

of d: from 0 to 39.5101, s: from 0 to 0.0377, m: from 0 to 1.2500

After quantization, the distribution still behaves in Gaussian distribution, so we can do quantization to the DMS space.

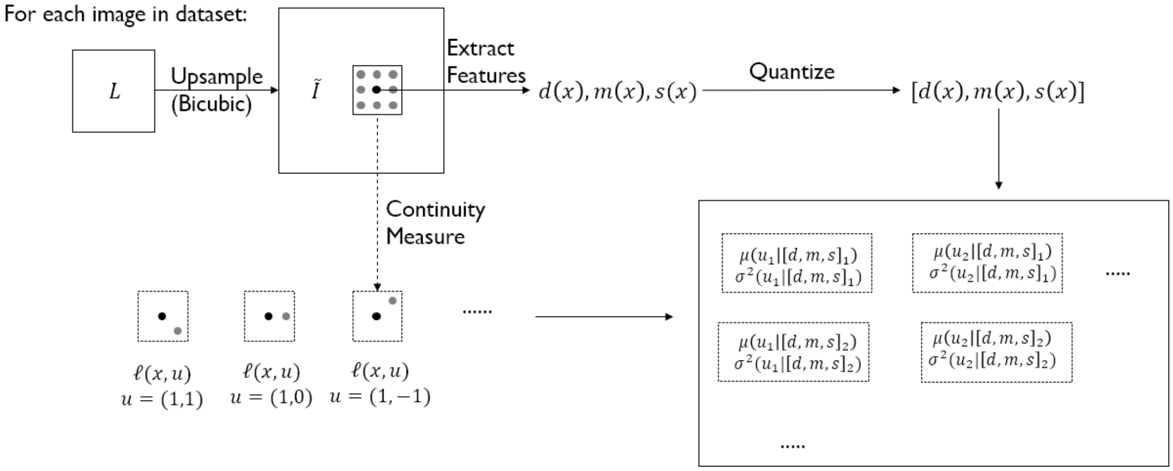
For each image in dataset:
L
Upsample (Bicubic)
$\tilde{I}$
Extract Features
$d(x), m(x), s(x)$
Quantize
$[d(x), m(x), s(x)]$
Continuity Measure
$\ell(x,u)$ $u=(1,1)$
$\ell(x,u)$ $u=(1,0)$
$\ell(x,u)$ $u=(1,-1)$
......
$\mu(u_1|[d,m,s]_1)$ $\sigma^2(u_1|[d,m,s]_1)$
$\mu(u_2|[d,m,s]_1)$ $\sigma^2(u_2|[d,m,s]_1)$
$\mu(u_1|[d,m,s]_2)$ $\sigma^2(u_1|[d,m,s]_2)$
$\mu(u_2|[d,m,s]_2)$ $\sigma^2(u_2|[d,m,s]_2)$
.....
.....

Figure 8: Training procedure of our implementation

Next, we proceed to the new image upsampling. Instead of solving the optimization problem in Raanan's paper, we find it simpler and quicker to predict the upsampled image by adding the averaged difference image over all u's to the temporary upsampled image, which is Bicubic interpolated image in our case.

Recall that the optimization problem in the paper was written in this form:

$$\max_I \log(P(I)) \quad s.t. \quad \mathfrak{D} \cdot I = L$$

This is equivalent to

$$\min_I \sum_u \frac{(\ell(x,u) \text{ -}\mu(u|[d(x),m(x),s(x)]))^2}{\sigma^2(u|[d(x),m(x),s(x)])} \quad s.t. \quad \mathfrak{D} \cdot I = L$$

$$where$$

$$\sum_u \frac{(\ell(x,u) \text{ -}\mu(u|[d(x),m(x),s(x)]))^2}{\sigma^2(u|[d(x),m(x),s(x)])}$$

$$= \frac{|u|}{|u|} \sum_u \frac{(\ell(x,u) \text{ -}\mu(u|[d(x),m(x),s(x)]))^2}{\sigma^2(u|[d(x),m(x),s(x)])}$$

$$\geq |u| \frac{\left(\sum_u \frac{\ell(x,u)}{|u|} \text{ - } \sum_u \frac{\mu(u|[d(x),m(x),s(x)])}{|u|}\right)^2}{\sum_u \frac{\sigma^2(u|[d(x),m(x),s(x)])}{|u|}}$$

Where the inequality is by Jensen's Inequality. Since the random variable here is $\ell(x,u)$, this is a convex function of $\ell(x,u)$.

The equality holds when $\sum_u \frac{\ell(x,u)}{|u|} = \sum_u \frac{\mu\left(u\middle|[d(x),m(x),s(x)]\right)}{|u|}$.

Finally, we can predict the upsampled image by adding the averaged difference image to the temporary upsampled image. Notice that we still need to guarantee that the downsampled image from the predicted image must be the same as the original one. We can generate a mask image indicating which pixel of the temporary upsampled image corresponds to the original image pixel. Then we can multiply this mask by the difference image so that we don't add difference values to the pixels which contains the original information.

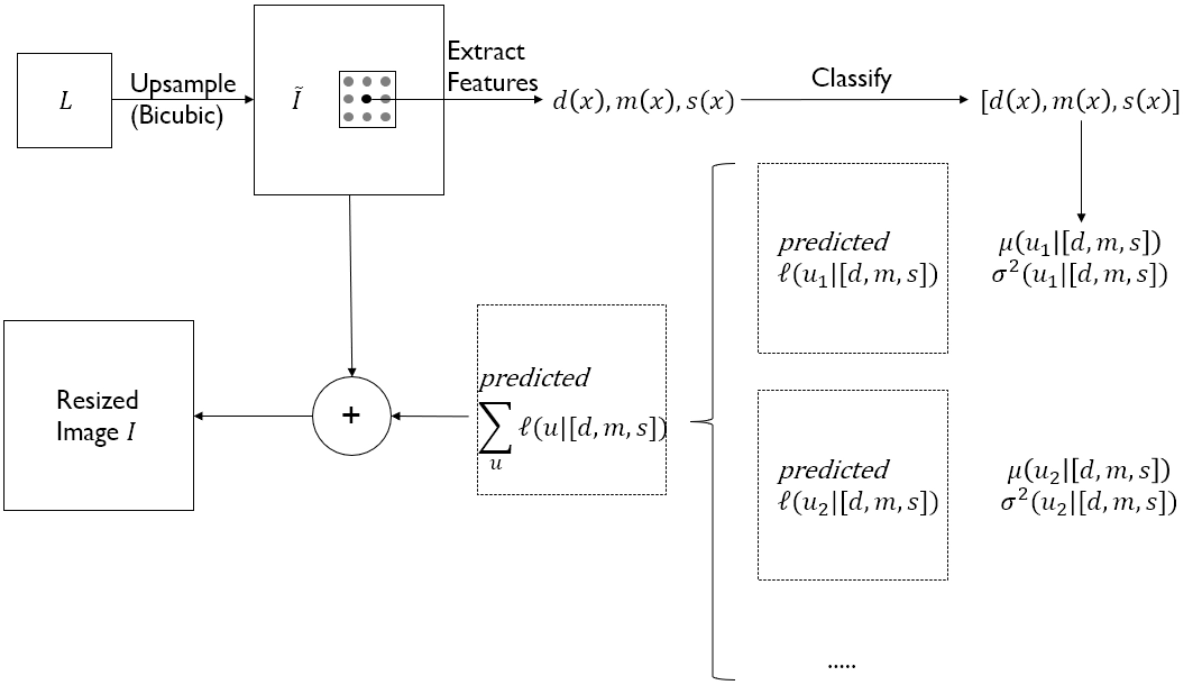

Figure 9: Testing procedure of our implementation

5 Numerical results

Since the difference between the Bicubic interpolated image and our prediction is not obvious, we can take a look at the difference images and the RMSE of these two images and the groundtruth.

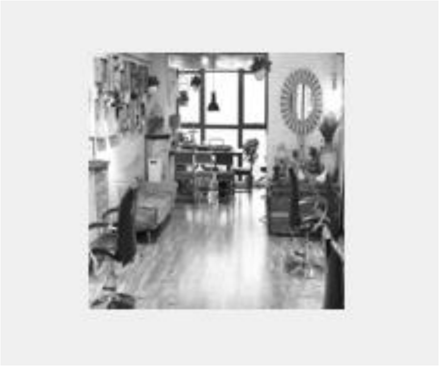

Figure 10: Original image of size 128x128

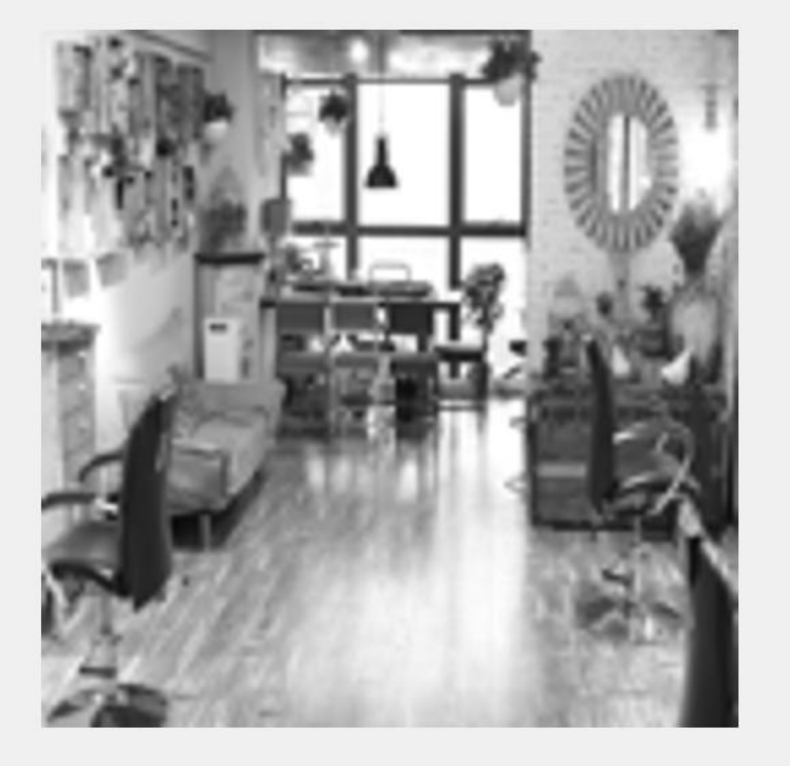

Figure 11: Upsampled image by Bicubic interpolation(512x512)

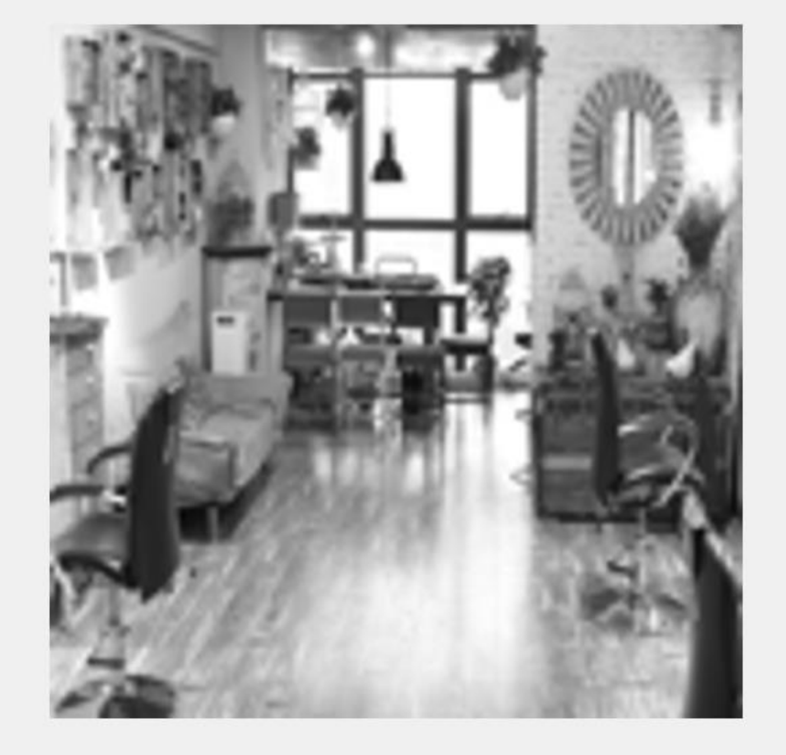

Figure 12: Upsampled image by our method(512x512)

Figure 13: Difference image (512x512)

The RMSE of the Bicubic-interpolated image = 0.1679.

The RMSE of our upsampled image = 0.1674.

To show the effect of edge impose, we can look at the test image of ‘cameraman.jpg’.

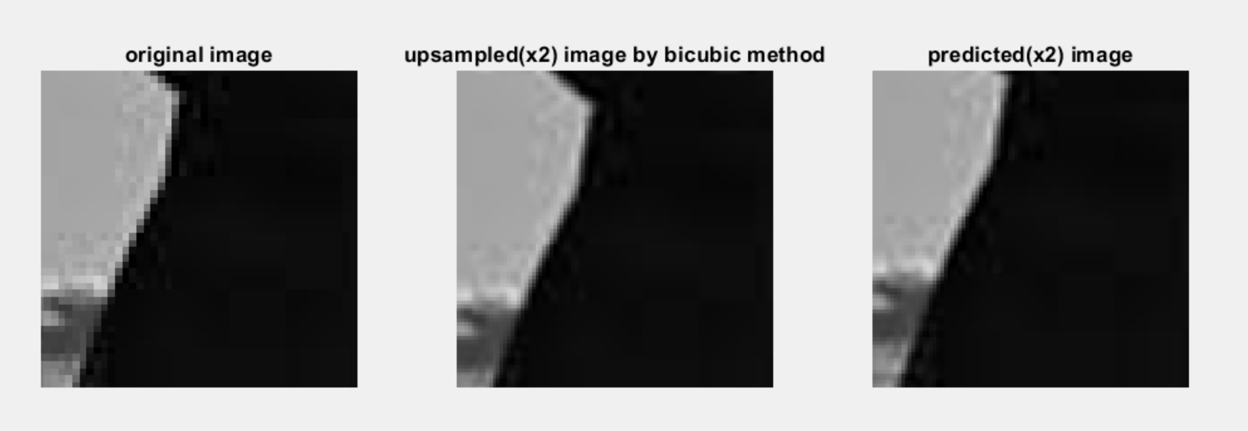

Figure 14: test on cameraman

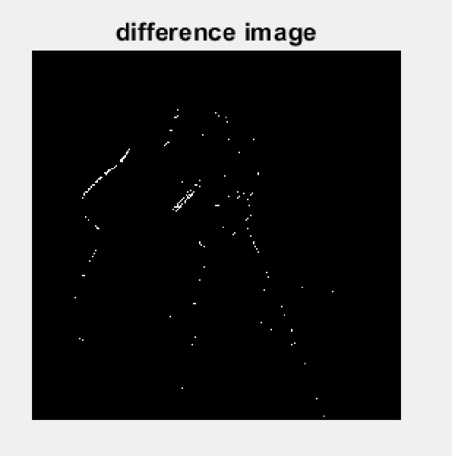

Figure 15: difference image between the upsampled image by bicubic method and our predicted image

Reference

[1] FATTAL, R. 2007. Image upsampling via imposed edge statistics. In *ACM transactions on graphics (TOG).* Vol. 26. No. 3.

[2]THVENAZ, P., BLU, T., AND UNSER, M. 2000. Image interpolation and resampling. In *Handbook of Medical Imaging, Processing and Analysis*, Academic Press, San Diego CA, USA, I. Bankman, Ed., 393–420.

[3] TUMBLIN, J., AND CHOUDHURY, P. 2004. Bixels: Picture samples with sharp embedded boundaries. In *Rendering Techniques*, 255–264.

[4] MORSE, B. S., AND SCHWARTZWALD, D. 2001. Image magni- fication using level-set reconstruction. In *Computer Vision and Pattern Recognition*, 333–340.

[5] CAPEL, D., AND ZISSERMAN, A. 1998. Automatic mosaicing with super-resolution zoom. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, Washington, DC, USA, 885

[6] FREEMAN, W. T., JONES, T. R., AND PASZTOR, E. C. 2002. Example-based super-resolution. In *IEEE Computer Graphics and Applications*, vol. 22, 56–65.

[7] OSHER, S., SOLE, A., AND VESE, L. 2003. Image decomposition and restoration using total variation minimization and the h −1 . *Multiscale Modeling & Simulation 1*, 3, 349–370.

[8] ALY, H., AND DUBOIS, E. 2005. Image up-sampling using total variation regularization with a new observation model. In *IEEE Transactions on Image Processing*, vol. 14, 1647–1659.

[9] HERTZMANN, A., JACOBS, C. E., OLIVER, N., CURLESS, B., AND SALESIN, D. H. 2001. Image analogies. In *ACM Transactions on Graphics*, ACM Press, New York, NY, USA, 327–340.

[10] BENZI, M., GOLUB, G. H., AND LIESEN, J. 2005. Numerical solution of saddle point problems. In *Acta Numer.*, vol. 14, 1– 137.

[11] HUANG, J., AND MUMFORD, D. 1999. Statistics of natural images and models. In *Computer Vision and Pattern Recognition*, 1541– 1547.