Image Super-Resolution and Sharpening via Least-Squares and Robust LOESS

Chih-Chun Ho (何智鈞) Shao-Chun Lu (呂紹群)

Advisors: Prof. Shih-Feng Shieh and Prof. Je-Chiang Tsai

June 2025

Contents

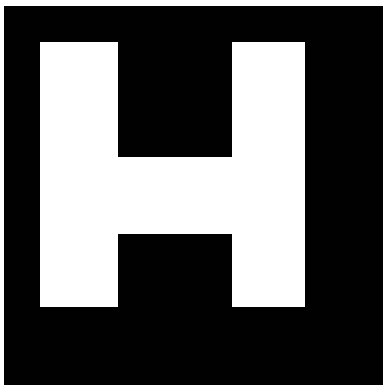
In this article, we will explain how our method works. Overall, we separate our method into three steps, which are

1. Apply Least Squares Method to double the size of an image.
2. Apply Adaptive Robust LOESS to smooth the super-resolved image.
3. Use the image obtained at step(1) minus the image obtained at step(2) to subtract those "sharp" parts. Find an optimal parameter α to multiply to those sharp parts and add back to the image obtained at step(2).

1 Least Squares Method

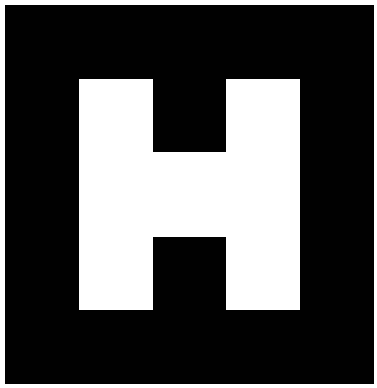
In this section, we will apply least squares method to deal with less-pixel images. First of all, we create a grayscale image with 100 pixels that displays "H".

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 255 & 255 & 1 & 1 & 1 & 255 & 255 & 1 & 1 \\ 1 & 255 & 255 & 1 & 1 & 1 & 255 & 255 & 1 & 1 \\ 1 & 255 & 255 & 1 & 1 & 1 & 255 & 255 & 1 & 1 \\ 1 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 1 & 1 \\ 1 & 255 & 255 & 255 & 255 & 255 & 255 & 255 & 1 & 1 \\ 1 & 255 & 255 & 1 & 1 & 1 & 255 & 255 & 1 & 1 \\ 1 & 255 & 255 & 1 & 1 & 1 & 255 & 255 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \Longrightarrow$$



And we compress the above matrix to a 5×5 matrix. The method we use to compress is delete the even-number columns and rows in the original 10×10 matrix. The compressed matrix and image are as follows:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 255 & 1 & 255 & 1 \\ 1 & 255 & 255 & 255 & 1 \\ 1 & 255 & 1 & 255 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \Longrightarrow$$



What we do on this page can be represented as the following matrix (*is what we want to reconstruct.).

$$\begin{bmatrix}
1 & * & 1 & * & 1 & * & 1 & * & 1 & * \\
* & * & * & * & * & * & * & * & * & * \\
1 & * & 255 & * & 1 & * & 255 & * & 1 & * \\
* & * & * & * & * & * & * & * & * & * \\
1 & * & 255 & * & 255 & * & 255 & * & 1 & * \\
* & * & * & * & * & * & * & * & * & * \\
1 & * & 255 & * & 1 & * & 255 & * & 1 & * \\
* & * & * & * & * & * & * & * & * & * \\
1 & * & 1 & * & 1 & * & 1 & * & 1 & * \\
* & * & * & * & * & * & * & * & * & *
\end{bmatrix}$$

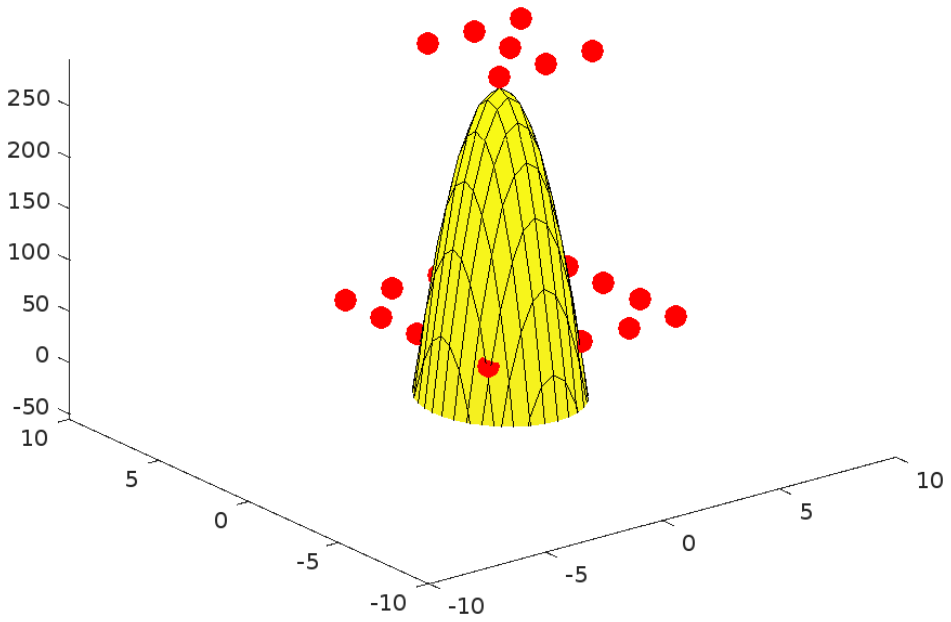
To fill those blank, we construct a function as follows.

$$\begin{array}{rccc} \varphi: & \mathbb{N}^2 \times \mathbb{R} & \longrightarrow & \mathbb{R}^3 \\ & (i,j,M_{ij}) & \longmapsto & (x,y,z) \end{array} \quad \text{i.e.,} \quad \varphi(i,j,M_{ij}) = (x,y,z)$$

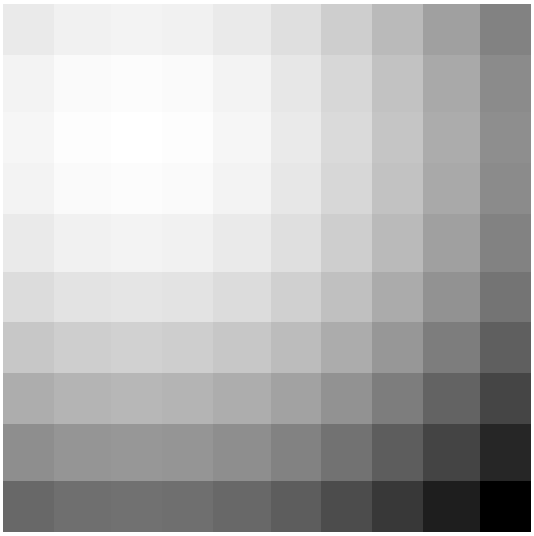
This function puts all data in the above matrix to $\mathbb{R}^3$, and we will find a **Quadratic Surface** ($p(x,y) = ax^2 + bxy + cy^2 + dx + ey + f$) to fit those data. In this case, there are 25 points but only six variables. As we learned in Linear Algebra, we apply " **least squares method**" to solve this problem. By the help of MATLAB, we have

$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} -36.2857 \\ -0.0000 \\ -29.0286 \\ 217.7143 \\ 174.1714 \\ -385.0800 \end{bmatrix}$$

And the graph in $\mathbb{R}^3$ is as follows.



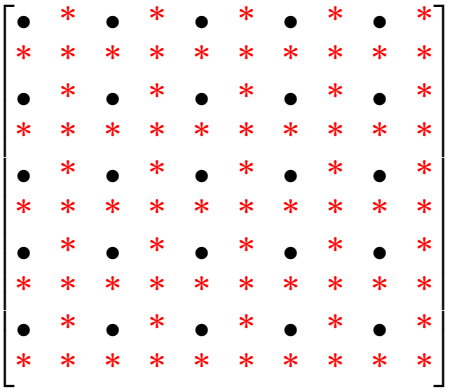
So the equation of $p(x, y)$ is known, then plug all blank pixels in the restored image to $p(x, y)$. The restored image is as follows.



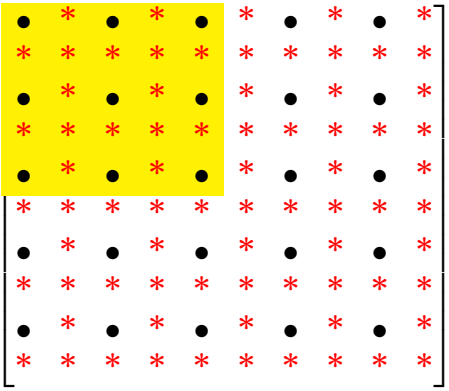
Over here, we have observed that we can use quadratic surface to deal with blank pixels when enlarging a less-pixel image. But apparently, we cannot use this method if the image size is large. The main reason is that when we apply the least square method to an image doubling a size 5×5, we actually solve a system with 25 constraints and only 6 variables. Similarly, when we apply the least squares method to double a size 50×50 image, we are actually solving a system with 2500 constraints and only 6 variables. No matter what method we use to solve this system, the solution is still awful. To address these challenges, we introduce the **Block Method**, which partitions the image into smaller, more manageable sections and also reduces the amount of constraints.

1.1 Block Method

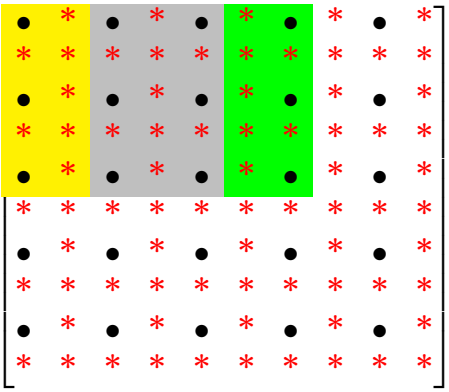
As an example on Page 3. We use • to replace numbers for a general case.



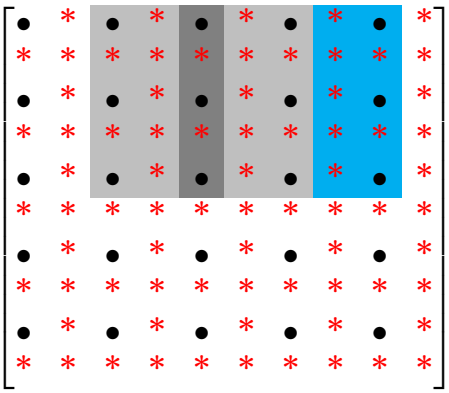
What we will do is slice the matrix into several small blocks and calculate the blank pixels individually.



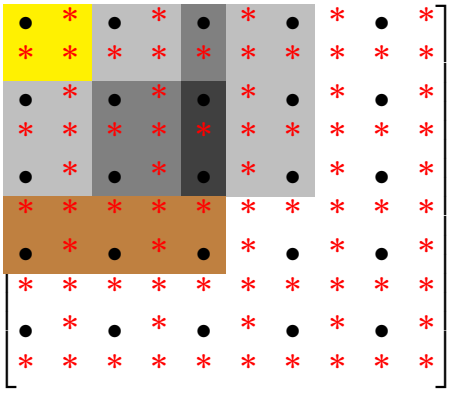
We apply Least Squares Method to the yellow region at above matrix. After we find those * values in yellow region, we let the block jump right **2** pixels.



The green region at above matrix is the next block which is no overlap with the yellow region. The gray region is the overlapping between the yellow region and green region. We deal with those overlapped regions by pretending we don't know that region is already calculated. After double counting, we save all the data we calculate and take their averages to determine the value they should take. Similarly, the next block should look like the matrix in next page.



We remove yellow region, which is unrelated to this step. Assuming we completed the first five rows. We then go back to the first block and jump down two pixels. What we do at this step is as follows.



Repeat the action above then we can complete the goal of doubling the size of an image. But note that the method of blocking is not unique, we can adjust the size of block to 3×3 or 7×7, or adjust the jumping pixel from 2 pixels to 1 pixel or 3 pixels. Also note that for the matrix at Page 6, if the size of the block becomes 7×7, then we will have 16 known points, which is sufficient to enhance the degree of equation to cubic. (The equation of Cubic Surface is $ax^3+bx^2+cx+dy^3+ey^2+fy+gx^2y+hxy^2+ixy+j$, which only has 10 coefficients to be determined.)
We fix jump pixels at 2 pixels and after we tried each block size we conclude that the best size of block is $\mathbf{5 \times 5}$.

2 Adaptive Robust LOESS

2.1 Robust LOESS

The following is a quick review of Robust LOESS. Consider $\{x_j, y_j\}_{j=1}^n$ which is a data sequence of length n. We will replace the original data sequence by $\{x_j, Y_j\}_{j=1}^n$. The procedure is as follows.
Suppose we want to replace the point (x_0, y_0) with $(x_0, \widehat{y_0})$.

1. Choose the window width h and determine the degree of polynomial.
   We use quadratic polynomial for example. Then the local polynomial fitting at x_i is as follows.

$$\mu(x_i) \approx \beta_0 + \beta_1(x_i - x_0) + \beta_2(x_i - x_0)^2$$

2. Let $u_i = \dfrac{|x_i - x_0|}{h}$ then the weight function at each x_i is defined as follows.

$$w_i(x_0) = \begin{cases} \left(1 - u_i^3\right)^3, & \text{if } u_i < 1, \quad \text{(i.e. all points in the window)} \\ 0, & \text{if } u_i \geq 1 \quad \text{(all points } \textbf{not} \text{ in the window)} \end{cases}$$

3. Then the objective function is as follows.

$$\begin{aligned} Q(\beta_0, \beta_1, \beta_2) &= \sum_{i=1}^n w_i(x_0) \left(y_i - \mu(x_i)\right)^2 \\ &= \sum_{i=1}^n w_i(x_0) \left[y_i - \left(\beta_0 + \beta_1(x_i - x_0) + \beta_2(x_i - x_0)^2\right)\right]^2 \end{aligned}$$

4. Suppose $(\widehat{\beta_0}, \widehat{\beta_1}, \widehat{\beta_2})$ minimize the above. Then y_0 will be replaced by $\widehat{\beta_0}$.

5. Repeat the above steps n times t obtain $\{x_j, \widehat{y_j}\}_{j=1}^n$.

Note that the above will be affected by the outlier. So the following procedure can reduce the influence of outlier.

6. Calculate the residual of all points. i.e. $r_i = y_i - \widehat{y_i}$.

7. The second weight function is defined as follows.

$$\delta_i = \begin{cases} \left[1 - \left(\dfrac{r_i}{6 \cdot \text{median}\left(|r_1|, |r_2|, \ldots, |r_n|\right)}\right)^2\right]^2, & \text{if } |r_i| < 6 \cdot \text{median}\,|r_i| \\ 0, & \text{if } |r_i| \geq 6 \cdot \text{median}\,|r_i| \end{cases}$$

8. Multiply $w_i(x_0)$ and δ_i to have final weight function $w_i^*(x_0)$

9. Return to step 3 and the final objective function is

$$Q(\beta_0, \beta_1, \beta_2) = \sum_{i=1}^n w_i^*(x_0) \left(y_i - \mu(x_i)\right)^2$$

10. Suppose $(\widetilde{\beta_0}, \widetilde{\beta_1}, \widetilde{\beta_2})$ minimize the above. Then y_0 will be replaced by $\widetilde{\beta_0}$.

11. Repeat the above then can obtain $\{x_j, Y_j\}_{j=1}^n$.

2.2 Adaptive Robust LOESS

In the last section, we have seen that we can use Robust LOESS to do 1-dimensional data sequence smoothing. However, we can actually treat every image as a 2-dimensional data sequence(x axis and y axis are the matrix index, and z axis is the color at each pixel). So in this part, we will try to generalize those formulas in Robust LOESS. Consider $\{x_j, y_j, z_j\}_{j=1}^n$ which is a 2-dimensional data sequence. We will replace the original data sequence by $\{x_j, y_j, Z_j\}_{j=1}^n$. The procedure is as follows.
Suppose we want to replace the point (x_0, y_0, z_0) with $(x_0, y_0, \widehat{z_0})$.

1. Choose the window radius R and the degree of polynomial.
   Suppose we use quadratic polynomial. Since this polynomial is in $\mathbb{R}^3$ space, so the local polynomial fitting will be the following.
   $$z = \mu(x_i, y_i) = \beta_0 + \beta_1(x_i - x_0) + \beta_2(y_i - y_0) + \beta_3(x_i - x_0)^2 + \beta_4(y_i - y_0)^2 + \beta_5(x_i - x_0)(y_i - y_0)$$
2. Let $u_i = \dfrac{\sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}}{R}$ then the weight function at each (x_i, y_i) is defined as follows.
   $$w_i(x_0, y_0) = \begin{cases} \left(1 - u_i^3\right)^3, & \text{if } u_i < 1, \quad \text{(i.e. all points in the circle)} \\ 0, & \text{if } u_i \geq 1 \quad \text{(all points } \textbf{not} \text{ in the circle)} \end{cases}$$
3. Then the objective function is as follows.
   $$Q(\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5) = \sum_{i=1}^{n} w_i(x_0, y_0) \left(z_i - \mu(x_i, y_i)\right)^2$$
4. Suppose $(\widehat{\beta_0}, \widehat{\beta_1}, \widehat{\beta_2}, \widehat{\beta_3}, \widehat{\beta_4}, \widehat{\beta_5})$ minimize the above. Then z_0 will be replaced by $\widehat{\beta_0}$.
5. Repeat the above steps then can obtain $\{x_j, y_j, \widehat{z_j}\}_{j=1}^n$.
6. Calculate the residual of all points. i.e. $r_i = z_i - \widehat{z_i}$.
7. The second weight function is defined as follows.
   $$\delta_i = \begin{cases} \left[1 - \left(\dfrac{r_i}{6 \cdot \text{median}\left(|r_1|, |r_2|, \ldots, |r_n|\right)}\right)^2\right]^2, & \text{if } |r_i| < 6 \cdot \text{median}\,|r_i| \\ 0, & \text{if } |r_i| \geq 6 \cdot \text{median}\,|r_i| \end{cases}$$
8. Multiply $w_i(x_0, y_0)$ and δ_i to have final weight function $w_i^*(x_0, y_0)$
9. Return to step 3 and the final objective function is
   $$Q(\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5) = \sum_{i=1}^{n} w_i^*(x_0, y_0) \left(z_i - \mu(x_i, y_i)\right)^2$$
10. Suppose $(\widetilde{\beta_0}, \widetilde{\beta_1}, \widetilde{\beta_2}, \widetilde{\beta_3}, \widetilde{\beta_4}, \widetilde{\beta_5})$ minimize the above. Then z_0 will be replaced by $\widetilde{\beta_0}$.
11. Repeat the above then can obtain $\{x_j, y_j, Z_j\}_{j=1}^n$.

3 Find those "Sharp" Parts

In the last two sections, we have seen that we can use Adaptive Robust LOESS to smooth the image. However, this action will cause the image to become blurred and erase those "sharp" parts. In this section, we will try to add those "sharp" parts back. Before start, we define some notation as follows,

$$\begin{cases} X : \text{Source Image(Size: } n \times n) \\ I : \text{Super-resolved Image} \\ B : \text{Image after Super-resolve and Adaptive Robust LOESS} \\ D = I - B : \text{Those "sharp" parts} \\ \widehat{I} : \text{Final Output} \end{cases}$$

By the definition of the notation, we can derive the following.

$$\widehat{I} = I + \alpha(I - B) = I + \alpha D \quad \alpha \in \mathbb{R}$$

(i.e. α boost those sharp parts and add back to the image, this will make the image sharper.)
Following is how we minimize the **Mean Squared Error** between $\widehat{I}$ and X.

$$\begin{aligned} MSE(\alpha) &= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\widehat{I}_{ij} - X_{ij}\right)^2 \\ &= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(I_{ij} + \alpha D_{ij} - X_{ij}\right)^2 = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left[(I_{ij} - X_{ij}) + \alpha D_{ij}\right]^2 \end{aligned}$$

Let $e_{ij} = I_{ij} - X_{ij}$, then

$$MSE(\alpha) = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left[(I_{ij} - X_{ij}) + \alpha D_{ij}\right]^2 = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(e_{ij} + \alpha D_{ij}\right)^2$$

Reshape e_{ij}, D_{ij} into a column vector of length n^2 and denoted by $\mathbf{e}$ and $\mathbf{d}$,then

$$\begin{aligned} MSE(\alpha) &= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(e_{ij} + \alpha D_{ij}\right)^2 = \frac{1}{n^2}\left\|\mathbf{e} + \alpha\mathbf{d}\right\|^2 = \frac{1}{n^2}\left(\mathbf{e} + \alpha\mathbf{d}\right)^{\mathrm{T}}\left(\mathbf{e} + \alpha\mathbf{d}\right) \\ &= \frac{1}{n^2}\left(\mathbf{e}^{\mathrm{T}}\mathbf{e} + \alpha\mathbf{e}^{\mathrm{T}}\mathbf{d} + \alpha\mathbf{d}^{\mathrm{T}}\mathbf{e} + \alpha^2\mathbf{d}^{\mathrm{T}}\mathbf{d}\right) = \frac{1}{n^2}\left(\mathbf{e}^{\mathrm{T}}\mathbf{e} + 2\alpha\mathbf{e}^{\mathrm{T}}\mathbf{d} + \alpha^2\mathbf{d}^{\mathrm{T}}\mathbf{d}\right) \end{aligned}$$

Basic Calculus for finding local minimum.

$$\frac{\mathrm{d}}{\mathrm{d}\alpha}MSE(\alpha) = \frac{2}{n^2}\left(\mathbf{e}^{\mathrm{T}}\mathbf{d} + \alpha\mathbf{d}^{\mathrm{T}}\mathbf{d}\right) = 0$$

Hence

$$\widehat{\alpha} = \frac{-\mathbf{e}^{\mathrm{T}}\mathbf{d}}{\mathbf{d}^{\mathrm{T}}\mathbf{d}} = \frac{-\sum_{i=1}^{n}\sum_{j=1}^{n} e_{ij}\cdot d_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{n}\left(D_{ij}\right)^2} = \frac{-\sum_{i=1}^{n}\sum_{j=1}^{n}\left(I_{ij} - X_{ij}\right)\left(I_{ij} - B_{ij}\right)}{\sum_{i=1}^{n}\sum_{j=1}^{n}\left(I_{ij} - B_{ij}\right)^2}$$

4 Advantage and Disadvantage

- Advantage : By the above three steps, we can obtain a super-resolved image and those edges and corners are sharpened. This will make the score of SSIM higher. (Higher SSIM score means a greater similarity between two images)
- Disadvantage
    1. Since the Adaptive Robust LOESS calculates more than $2n^2$ times for a size $n \times n$ image, our algorithm takes a lot of time.
    2. As the algorithm shows, our method relies on the reference image. However, in most cases of super-resolution imaging, there is no high-resolution source image.

References

[1] Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, *74*(368), 829–836. https://doi.org/10.1080/01621459.1979.10481038