Super-Resolution Reconstruction Without Explicit Subpixel Motion Estimation

Xilei Wang*
Stanford University
450 Serra Mall, Stanford, CA 94305
xileiw@stanford.edu

Xiao Ju*
Stanford University
450 Serra Mall, Stanford, CA 94305
xju@stanford.edu

Abstract

Super resolution images are widely used in many fields. Classic super-resolution techniques generally involve fusing a number of low-resolution images with sub-pixel motion difference and correct estimation of the motion. This kind of accurate estimate, however, is not always available. Incorrect estimation of the motion generally produces undesirable results. In this report, two super-resolution reconstruction (SRR) algorithms without explicit subpixel motion estimation are presented. When given a sequence of low-resolution video frames, a sequence of high-resolution images can be produced. The first SRR [7] relies on classic SRR formulation and probablistic motion estimation. The second SRR [9] is based on 3-dimentional Steering Kernel Regression that captures spatiotemporal information around neighbourhoods. The concept and model is first introduced. The resulting images using the two algorithms and other methods are them compared and analyzed. Finally, comparation between these two algorithms is discussed.

1. Introduction

Super Resolution images are nowadays widely used in areas such as surveillance, forensic, scientific, medical and video [1]. In order to meet this increasing demand, researchers delved into the area of super-resolution reconstruction (SRR). The technique of constructing a super-resolution image from one or a sequence of low-resolution images, has already been widely applied.

Generally, SRR algorithms reconstruct the target images from a sequence of similar low-resolution image by fusing them. Classically, SRR techniques generally require that all images in the sequence must have a sub-pixel motion from some of other images, and the correct estimation of this motion must be available. In some cases, such as taking some scientific photos on things that does not move or move slowly, this requirement is not very demanding since there are available ways for taking such photo sequence. However, in areas such as video making, where things in the images are seldom move to one direction altogether at the same speed, it is highly unlikely that this kind of images and estimation are available. In classical SRR algorithm, incorrect estimation of the motion generally produces blurry results, which is clearly undesirable in most applications [8].

Therefore, the possibility to reconstruct super-resolution image from low-resolution image without sub-pixel motion estimation or movement is one of the main concerns in the field of super-resolution imaging. In this project, two super-resolution imaging algorithms without requirements for sub-pixel motion estimation were explored and implemented, namely, Super Resolution with Probabilistic Motion Estimation [7] and 3-D Iterative Steering Kernel Regression (3-D ISKR) algorithms [9].

The rest of this report is organized as follows. Section 2 shows a number of related works in SRR. The two methods that were implemented are shown in Section 3. Results of implementation and experiment are discussed in Section 4. The conclusion of this project is given in Section 5.

2. Related Work

Glasner et. al [6] proposed a unified computational framework that combines Classical SRR and Example-based SRR to obtain a super-resolution image from a single low-resolution image. Recurrence of patches within the same image scale provides basis for applying Classical SRR constraints. For Example-based SRR, the low-resolution/high-resolution patch correspondences can be learned from patch recurrence across multiple image scales of a single image, without any additional external information.

In [10], Zhang et. al proposed a maximum a posteriori probability (MAP) Super Resolution framework that incorporates Non-local Means (NLM) and Steering Kernel Regression (SKR) prior terms to obtain a super-resolution image from a low-resolution image. NLM prior considers redundancy of patches in images and SKR assumes that a target pixel is a weighted average from its neighbors. Super-resolution image is iteratively optimized using the gradient

descent method.

Elad et. al reviewed [5] a class of approaches for this problem, namely, Exampled-based approach. In their work, this class of approaches is further divided into three sub-classes. First, using examples to fine-tune the parameters of previously defined regularization expressions. Second, using them directly for the reconstruction procedure. Last but not least, combination of approaches in first and second sub-class.

In [4], Danielyan et. al. proposed a framework using an algorithm known as block-matching and 3D filtering (BM3D) [3] to compensate the lack of motion information in super-resolution reconstruction. The proposed framework could be applied to super-resolution reconstruction problem for both video and image reconstruction.

3. Theory

In this section, two super-resolution reconstruction algorithms that do not require accurate, explicit sub-pixel estimation are presented. Section 3.1 introduces SRR with Probability Motion Estimation proposed by [7]. Section 3.2 describes SRR with 3-D Steering Kernel Regression proposed by [9].

3.1. SRR with Probability Motion Estimation

Given a sequence of low resolution vectorized images $\{\mathbf{y}\}_{t=1}^{T} \in \mathbb{R}^N$, scaling factor s and target sequence of high resolution vectorized images $\{\mathbf{x}\}_{t=1}^{T} \in \mathbb{R}^{sN}$, SSR algorithms with explicit motion estimation typically relate above two terms by following equation:

$$\mathbf{y}_t = \mathbf{DHF}_t\mathbf{x} + \mathbf{n}_t \qquad \text{for } t = 1, 2, \ldots, T, \tag{1}$$

where $\mathbf{F}_t \in \mathbb{R}^{sN \times sN}$ is a bijective mapping matrix describes the motion translation; $\mathbf{H} \in \mathbb{R}^{sN \times sN}$ is a toeplitz matrix equivalently as convoluting with a spatially-invariant blur kernel h; $\mathbf{D} \in \mathbb{R}^{N \times sN}$ is a decimation matrix that downsamples the high resolution vectorized image to low resolution vectorized image based on the scaling factor; $\mathbf{n}_t \in \mathbb{R}^N$ is the additive zero-mean Gaussian noise. Typically, $\mathbf{y}_1$ is referred as the reference image with $\mathbf{F}_1 = \mathbf{I}$. Using maximum-likelihood estimation, $\mathbf{x}$ is obtained by solving the least-square problem

$$L_{ML}(\mathbf{x}) = \frac{1}{2}\sum_{t=1}^{T} \|\mathbf{DHF}_t\mathbf{x} - \mathbf{y}_t\|_2^2. \tag{2}$$

Setting the derivative of Eq. 2 with respect to $\mathbf{x}$ to 0

$$\frac{\partial L_{ML}(\mathbf{x})}{\partial \mathbf{x}} = \sum_{t=1}^{T} \mathbf{F}_t^T\mathbf{H}^T\mathbf{D}^T(\mathbf{DHF}_t\mathbf{x} - \mathbf{y}_t) = 0, \tag{3}$$

leading to a linear system $\mathbf{A}\hat{\mathbf{x}}_{ML} = \mathbf{b}$, where $\mathbf{A} = \sum_{t=1}^{T} \mathbf{F}_t^T\mathbf{H}^T\mathbf{D}^T\mathbf{DHF}_t$ and $\mathbf{b} = \sum_{t=1}^{T} \mathbf{F}_t^T\mathbf{H}^T\mathbf{D}^T\mathbf{y}_t$.

In many cases, insufficient measurements lead to ill-conditioned $\mathbf{A}$. Thus, a regularized least-square is often applied

$$L_{RML}(\mathbf{x}) = \frac{1}{2}\sum_{t=1}^{T} \|\mathbf{DHF}_t\mathbf{x} - \mathbf{y}_t\|_2^2 + \lambda \cdot TV(\mathbf{x}), \tag{4}$$

where $TV(\mathbf{x})$ denotes the total variation of $\mathbf{x}$. This problem can effectively solved by the alternating direction method of multipliers(ADMM) [2].

Minimizing above problems relies on the acknowledge of $\mathbf{D}$, $\mathbf{H}$ and $\mathbf{F}_t$, particularly of $\mathbf{F}_t$ with a sub-pixel accuracy of motion estimation. In practice, such accuracy is hard to achieve and classical SRR algorithms usually assume simple motion like pure axial shift or global affine wrap. For general motion, this assumption is not sufficient and inaccurate motion estimation often results annoying artifacts.

In order to circumvent explicit motion estimation, Matan proposed a probabilistic motion estimation model where each pixel in target image is represented as a weighted sum of possible corresponding pixels in low resolution images. Assume the maximal axial shift is D pixels, then a set of $M = (2D+1)^2$ displacements covers all the possible translations. Letting $\{\mathbf{F_m}\}_{m=1}^{M}$ denote the translation matrices for the set of all displacements, a bijective mapping matrix is decomposed as following weighted sum

$$\mathbf{F}_t\mathbf{x} = \sum_{m=1}^{M} \mathbf{Q}_{m,t}\mathbf{F}_m\mathbf{x}, \tag{5}$$

where $\{\mathbf{Q}_{m,t}\}_{m=1}^{M}$ are diagonal weight matrices with 1-es for pixels whose motion is the displacement $\mathbf{F}_m$ and zeros otherwise. Slightly diverting from this approach, a simplified version leads to following probabilistic maximum likelihood loss function

$$L_{PML}(\mathbf{x}) = \frac{1}{2}\sum_{m=1}^{M}\sum_{t=1}^{T} \|\mathbf{W}_{m,t}\mathbf{DHF}_m\mathbf{x} - \mathbf{y}_t\|_2^2, \tag{6}$$

which accumulates the squared errors that directly resulted from all possible global displacements, with a weight matrices $\mathbf{W}_{m,t}$.

Optimizing Eq. 6 can be splitted into two steps: the fusion step

$$\underset{\mathbf{z}}{\text{minimize}} \quad L_{fs}(\mathbf{z}) = \frac{1}{2}\sum_{m=1}^{M}\sum_{t=1}^{T} \|\mathbf{W}_{m,t}\mathbf{DF}_m\mathbf{z} - \mathbf{y}_t\|_2^2, \tag{7}$$

the deblurring step

$$\underset{\mathbf{x}}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{Hx} - \mathbf{z}\|_2^2 + \lambda \cdot TV(\mathbf{x}). \tag{8}$$

This two-step process is sub-optimal to the joint optimization, but nevertheless leads to a simplified algorithm. The fusion step is solved by a pixel-wise update and the deblurring step is effectively solved by ADMM.

3.1.1 Pixel-wise Update for Fusion Step

Setting the derivative of Eq. 7 to 0

$$\frac{\partial L_{fs}(\mathbf{z})}{\partial \mathbf{z}} = \sum_{m=1}^{M}\sum_{t=1}^{T} \mathbf{F}_m^T \mathbf{D}^T \mathbf{W}_{m,t}(\mathbf{D}\mathbf{F}_m\mathbf{z} - \mathbf{y}_t) = 0, \quad (9)$$

and introducing new notations

$$\widetilde{\mathbf{W}}_m = \sum_{t=1}^{T} \mathbf{W}_{m,t}, \quad \widetilde{\mathbf{y}}_m = \sum_{t=1}^{T} \mathbf{W}_{m,t}\mathbf{y}_t, \quad (10)$$

the minimization problem converts to solving a linear equation

$$\left[\sum_{m=1}^{M} \mathbf{F}_m^T \mathbf{D}^T \widetilde{\mathbf{W}}_m \mathbf{D}\mathbf{F}_m\right]\mathbf{z} = \sum_{m=1}^{M} \mathbf{F}_m^T \mathbf{D}^T \widetilde{\mathbf{y}}_m. \quad (11)$$

Following analysis using 2D indexing for vectorized images. On the right-hand-side, $\mathbf{F}_m^T\mathbf{u}$ acts as the inverse displacement which maps pixel $[i+dx(m), j+dy(m)]$ of $\mathbf{u}$ to location $[i,j]$; $\mathbf{u} = \mathbf{D}^T\widetilde{\mathbf{y}}_m$ scales up $\widetilde{\mathbf{y}}_m$ with zero-fillings. In specific,

$$\mathbf{u}[i+dx(m), j+dy(m)] = \begin{cases} \widetilde{\mathbf{y}}_m[k,l] & \text{if } [k,l] = [i+dx(m), j+dy(m)]/s \\ 0 & \text{otherwise} \end{cases}. \quad (12)$$

Define the following set

$$N(i,j) = \{[k,l] | \forall m \in [1,M], s\cdot k = i + dx(m), \; s\cdot l = j + dy(m)\}, \quad (13)$$

the right-hand-side can be simplified to the following weighted sum

$$\text{RHS}[i,j] = \sum_{[k,l]\in N(i,j)} \sum_{t=1}^{T} \mathbf{W}_{m,t}[k,l]\mathbf{y}_t[k,l]. \quad (14)$$

On the left-hand-side, the operator $\mathbf{F}_m^T\mathbf{D}^T\widetilde{\mathbf{W}}_m\mathbf{D}\mathbf{F}_m$ shifts pixel $[i,j]$ to location $[i+dx(m), j+dy(m)]$, nulls or weights it based on whether $[i+dx(m), j+dy(m)]/s$ is an integer, finally shifts the outcome back by $[-dx(m), -dy(m)]$ to its original location $[i,j]$. Because every output pixel only depends on the pixel at the same location of input, the left-hand-side can be simplified to

$$\text{LHS}[i,j] = \sum_{[k,l]\in N(i,j)} \sum_{t=1}^{T} \mathbf{W}_{m,t}[k,l]\mathbf{z}[i,j]. \quad (15)$$

Combining Eq. 14 and Eq. 15, a closed form expression for pixel $[i,j]$ of the estimated $\mathbf{z}$ is given,

$$\hat{\mathbf{z}}[i,j] = \frac{\sum\limits_{[k,l]\in N(i,j)} \sum_{t=1}^{T} \mathbf{W}_{m,t}[k,l]\mathbf{y}_t[k,l]}{\sum\limits_{[k,l]\in N(i,j)} \sum_{t=1}^{T} \mathbf{W}_{m,t}[k,l]}. \quad (16)$$

To complete the algorithm, the definition of weight matrix is given as follows

$$\mathbf{W}_{m,t}[i,j] = \exp\left\{-\frac{\|\mathbf{R}_{i,j}(\mathbf{D}\mathbf{F}_m\mathbf{z} - \mathbf{y}_t\|_2^2}{2\sigma^2}\right\} \cdot f\Big(\sqrt{(dx(m))^2 + (dy(m))^2 + (t-1)^2}\Big), \quad (17)$$

where $\mathbf{R}_{i,j}$ extracts $q \times q$ local patch centered at pixel $[i,j]$. The first term is inversely proportional to the Euclidean distance between the translated image and the input image within the local support. The second term enforces a decaying factor as a function of displacement and time shift with respect to the reference frame. The function f can be any monotonically non-increasing function. In experiments, linear function, exponential function and Gaussian function with different varying parameters were tested. Please refer to section 4.1 for detail analysis.

In practice, as $\mathbf{z}$ is unknown at the beginning, $\mathbf{W}_{m,t}$ is computed by replacing $\mathbf{z}$ with a linear interpolated $\mathbf{y}_1$. In the next iteration, $\mathbf{W}_{m,t}$ is recomputed using the estimated $\hat{\mathbf{z}}$ from last iteration. In experiments, two such iterations is empolyed to obtain the final estimation.

3.1.2 ADMM-based Deblurring Step

In ADMM notation, the TV-regularized deconvolution problem is formulated as

$$\underset{\mathbf{x}}{\text{minimize}} \; \underbrace{\frac{1}{2}\|\mathbf{H}\mathbf{x} - \mathbf{z}\|_2^2}_{f(\mathbf{x})} + \underbrace{\lambda\|\mathbf{k}\|_1}_{g(\mathbf{k})} \quad \text{subject to } \mathbf{D}\mathbf{x} - \mathbf{k} = 0, \quad (18)$$

where

$$\mathbf{D}_x\mathbf{x} = \text{vec}\,(d_x * x), \quad d_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad (19)$$

$$\mathbf{D}_y\mathbf{x} = \text{vec}\,(d_y * x), \quad d_y = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad (20)$$

$$\mathbf{D} = [\mathbf{D}_x, \mathbf{D}_y]^T. \quad (21)$$

The operator $\text{vec}(\cdot)$ vectorizes a 2D image, $x \in R^{\sqrt{sN}\times\sqrt{sN}}$ is the 2D version of $\mathbf{x}$, $\mathbf{k} \in \mathbb{R}^{2sN}$ is the

stacked and vectorized image gradient along x and y axis. Clearly, this formulation is equivalent to the original problem. ADMM splits the objective into a weighted sum of two independent functions $f(\mathbf{x})$ and $g(\mathbf{k})$ that are only linked through the constraints.

Following the general ADMM strategy, the Augmented Lagrangian of Eq. 18 is formed as

$$L_\rho(\mathbf{x},\mathbf{k},\mathbf{y}) = f(\mathbf{x}) + g(\mathbf{k}) + \mathbf{y}^T(\mathbf{D}\mathbf{x}-\mathbf{k}) + \frac{\rho}{2}\|\mathbf{D}\mathbf{x}-\mathbf{k}\|_2^2 \tag{22}$$

Using the scaled form of the Augmented Lagrangian, the following iterative updates rules can be derived:

$$\mathbf{x} \leftarrow \mathbf{prox}_{f,\rho}(\mathbf{v}) = \arg\min_{\mathbf{x}} f(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{D}\mathbf{x}-\mathbf{v}\|_2^2, \quad \mathbf{v} = \mathbf{k}-\mathbf{u} \tag{23}$$

$$\mathbf{k} \leftarrow \mathbf{prox}_{g,\rho}(\mathbf{v}) = \arg\min_{\mathbf{k}} g(\mathbf{k}) + \frac{\rho}{2}\|\mathbf{v}-\mathbf{k}\|_2^2, \quad \mathbf{v} = \mathbf{D}\mathbf{x}+\mathbf{u} \tag{24}$$

$$\mathbf{u} \leftarrow \mathbf{v} + \mathbf{D}\mathbf{x} - \mathbf{k}, \tag{25}$$

where $\mathbf{u} = (1/\rho)\mathbf{y} \in \mathbb{R}^{2sN}$.

For the $\mathbf{x}$-update, the inverse filtering proximal operator is given

$$\mathbf{prox}_{f,\rho}(\mathbf{v}) = \mathcal{F}^{-1}\{\mathcal{F}\{h\}^* \cdot \mathcal{F}\{z\} + \rho(\mathcal{F}\{d_x\}^* \cdot \mathcal{F}\{v_1\} + \mathcal{F}\{d_y\}^* \cdot \mathcal{F}\{v_2\}) ./\mathcal{F}\{h\}^* \cdot \mathcal{F}\{h\} + \rho(\mathcal{F}\{d_x\}^* \cdot \mathcal{F}\{d_x\} + \mathcal{F}\{d_y\}^* \cdot \mathcal{F}\{d_y\})\}, \tag{26}$$

where $\mathbf{v} = \mathbf{D}\mathbf{x} + \mathbf{u}$, $\cdot$ is the element-wise product, $z \in R^{\sqrt{sN}\times\sqrt{sN}}$ is the matrix version of $\mathbf{z}$, $v_1 \in R^{\sqrt{sN}\times\sqrt{sN}}$ and $v_2 \in R^{\sqrt{sN}\times\sqrt{sN}}$ are the 2D image gradient along x and y axis, $\mathcal{F}$ and $\mathcal{F}^{-1}$ are the 2D Fourier and inverse Fourier operator.

For the $\mathbf{k}$-update, the ℓ_1 proximal operator exists as

$$\mathbf{prox}_{f,\rho}(\mathbf{v}) = \mathcal{S}_{\lambda/\rho}(\mathbf{v}), \tag{27}$$

with $\mathbf{v} = \mathbf{D}\mathbf{x} + \mathbf{u}$ and $\mathcal{S}_{\lambda/\rho}(\cdot)$ being the element-wise soft thresholding operator

$$\mathcal{S}_\kappa(v) = \begin{cases} v-\kappa & v > \kappa \\ 0 & |v| \le \kappa \\ v+\kappa & v < -\kappa \end{cases} = (v-\kappa)_+ - (-v-\kappa), \tag{28}$$

which can be implemented effciently.

3.2. SRR with 3-D Steering Kernel Regression

First, the concept of kernel regression is intruduced in this section. Then, 3-D Steering Kernel Regression proposed by [9] is explained. The algorithm is in space-time and does not require explicit accurate sub-pixel estimation. Finally, An iteration version of the algorithm [9] is presented.

3.2.1 Kernel Regression

Kernel regressin model is defined as

$$y_i = z(x_i) + \epsilon_i, i = 1, ..., P, x_i = [x_{1i}, x_{2i}]^T \tag{29}$$

where y_i is a nosiy sample, $z(\cdot)$ is the regression function to be estimated, ϵ_i is an i.i.d zero mean noise, P is the number of sample pixels around interest pixel x. If x is near sample pixel x_i, the Nth order Taylor series is

$$\begin{aligned} z(x_i) &\approx z(x) + \{\nabla z(x)\}^T(x_i - x) \\ &\quad + \frac{1}{2}(x_i-x)^T\{Hz(x)\}(x_i-x) + ... \\ &= \beta_0 + \beta_1^T(x_i - x) \\ &\quad + \beta_2^T vech\{(x_i-x)(x_i-x)^T\} + ... \end{aligned} \tag{30}$$

where ∇ and H are the gradient and Hessian operators, and $vech(\cdot)$ is the half-vectorizatino operator. The least- square fomulation is

$$\begin{aligned} &\min_{\{\beta_n\}_{n=0}^N} \sum_{i=1}^P [y_i - \beta_o - \beta_1^T(x_i - x) \\ &- \beta_2^T vech\{(x_i-x)(x_i-x)^T\} - ...]^2 K_{H_i}(x_i - x) \end{aligned} \tag{31}$$

where

$$K_{H_i}(x_i - x) = \frac{1}{det(H_i)} K(H_i^{-1}(x_i - x)) \tag{32}$$

N is regression order, $K(\cdot)$ is kernel function and H_i is so-moothing matrix that indicates footprint of the kernel function. Nadaraya-Waston estimator (for zero orde regression) can be expressed as

$$\hat{z}(x) = \hat{\beta}_0 = \frac{\sum_{i=1}^P K_{H_i}(x_i - x) y_i}{\sum_{i=1}^P K_{H_i}(x_i - x)} \tag{33}$$

The classic kernel regression generates an estimator that is a linear combination of neighboring sample pixels. For the next section, a nonlinear steering kernel regression is introduced.

3.2.2 3-D Steering Kernel Regression

3-D Steering Kernel Regression is a Space-Time regression.The first two dimensions represent spatial coordinates and the third dimension represents temporal coordinate. Thus it can be used for processing video data, which is a continuous sequence of frames. The smoothing matrix is defined as a symmetric matrix

$$H_i^s = hC_i^{-\frac{1}{2}} \tag{34}$$

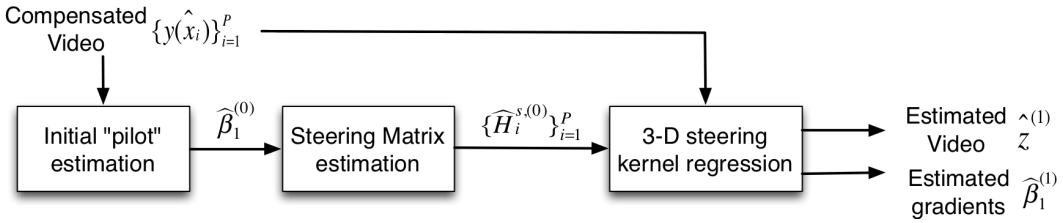

Figure 1. Steering kernel regression, initialization step.

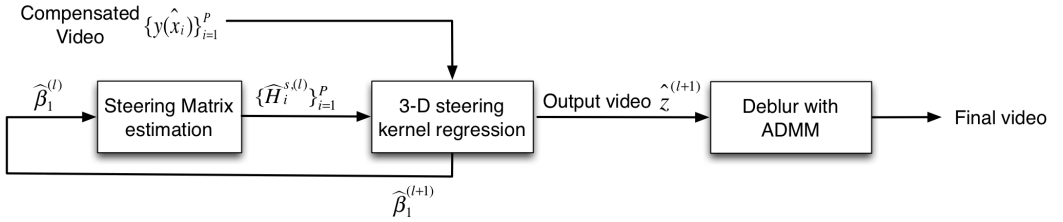

Figure 2. Steering kernel regression, iteration step.

which is called steering matrix. The matrix C_i can be estimated as local covariance of neighboring sptial gradient vectors by

$$\hat{C}_i = J_i^t J_i \tag{35}$$

with

$$J_i = \begin{bmatrix} z_{x_1}(x_1) & z_{x_2}(x_1) & z_t(x_1) \\ \vdots & \vdots & \vdots \\ z_{x_1}(x_P) & z_{x_2}(x_P) & z_t(x_P) \end{bmatrix} \tag{36}$$

where $z_{x_1}(\cdot)$,$z_{x_2}(\cdot)$ and $z_t(\cdot)$ are the first derivatives along x_1, x_2 and t axes. Then based on [9], 3-D steering kernel function is defined as

$$K_{H_i^n}(x_i - x) = \sqrt{\frac{det(C_i)}{(2\pi h^2)^3}} exp\{-\frac{1}{2h^2}\|c_i^{1/2}(x_i - x)\|_2^2\}$$

$$x = [x_1, x_2, t]^T \tag{37}$$

The kernel weights are calculated based on local window, thus captures local signal structure and is a non-linear regression. Since the kernel is 3-D, it contains both spatial and temporal information.

3.2.3 Iteration Version of Algorithm

Iteration version of 3-D ISKR is shown in Fig. 1 and Fig. 2. It is implemented as follows.

A. Motion Compensation

Computing a compensated video sequence is required before implement 3-D ISKR. The reason is that when there is large motion, similar pixels are found far away in space, even though they are close in time dimension. If these displacements are cancelled out, the estimated steering kernel will spread across neighboring frames and take advantage of information in space-time neighborhood.

B. Initialization Step

With compensated video sequence, the gradients $\hat{\beta}_1^{(0)}$ at the positions $\hat{x}_i{}_{i=1}^{P}$ is computed. This process is showed as "pilot estimation" in Fig. 1. Then steering matrix $\hat{H}_i^{s,(0)}$ is created by Eq. 34. Finally, they are plugged into Eq. 37 to estimate both $z(x)$ at interest pixel x and its gradients $\hat{\beta}_1^{(1)}$.

C. Iteration Step

Using $\hat{\beta}_1^{(1)}$, steering matrix $\hat{H}_i^{s,(1)}$ is created. Then, steering kernel regression is applied to input compensated video. Gradients $\hat{\beta}_1^{(2)}$ is estimated and next iteration begins. At last, the video is deblurred with ADMM as explained in Section 3.1.2.

4. Results and Analysis

In this section, super-resolution reconstruction results are shown and analyzed both visually and statistically. For SRR with Probability Motion Estimation, both fusion and deblur step were implemented and different f-fuction were tried. For SRR with 3-D ISKR, partial code from `https://users.soe.ucsc.edu/~milanfar/software/` was used and deblur with ADMM was implemented.

Figure 3. Results for the 8th, 18th and the 23th frames from Suzie sequence. From left to right: Low quality, Linear Interpolation, SRR with Probability Motion Estimation using linear function, exponential function and Gaussian function in 3 different σs.

Figure 4. Detail of the 8th frame from "Suzie" Sequence. From left to right: Low quality, Linear Interpolation, SRR with Probability Motion Estimation using Gaussian function.

4.1. SRR with Probability Motion Estimation

Fig. 3 shows the result of our implementation of this algorithm and Fig. 4 shows the detail.

In Fig. 3, each row shows the result of one frame in the video. Three frames selected here are Frame 8, 18 and 23. The leftmost column shows the original low-resolution image, the second column shows the result of the simple interpolation algorithm. The rest of the columns show the results of our implementation with different f-function or parameters of f-function.

As discussed in Section 3.1, the weight of each pixel is calculated with an f-function, which should be a monotonically non-increasing function. In order to find the best performing result, a few possible f-functions were tried and their corresponding results are shown in Fig. 3. Each method are further discussed as follows.

The first simple function tried was the linear functions with negative gradient. In other words,

$$y = kx + b(k < 0) \tag{38}$$

In this kind of f-function, there are two parameters, the gradient k and the intersection with y-axis b. Fig. 3 shows the optimized result of each frame. The Peak Signal to Noise Ratio (PSNR) curves vs both k and b for frame 8 is shown in Fig. 5.

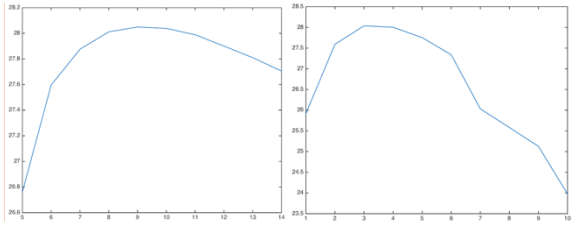

Figure 5. Linear Function (a)PSNR Vs k (b)PSNR Vs b

In Fig. 5(a), the value of b is fixed at 10 and the value of k varies from -1 to -10 (the x-axis in the figure shows the value of k instead of k). Similarly in Fig. 5(b), k is fixed at -3, while the value of b varies from 5 to 14. As shown in the figure, the combination of k and b that gives the best result is k = -3 and b = 9. In fact, among all f-functions that have been implemented, the one that generally perform best is this simple linear function and Gaussian Function (will be discussed a few paragraphs later).

The second f-function implemented was the exponential

Figure 6. Results for the 4th, 8th and the 21th frames from Foreman sequence. From left to right: Low Quality, Linear Interpolation, SRR with Probability Motion Estimation and 3-D ISKR.

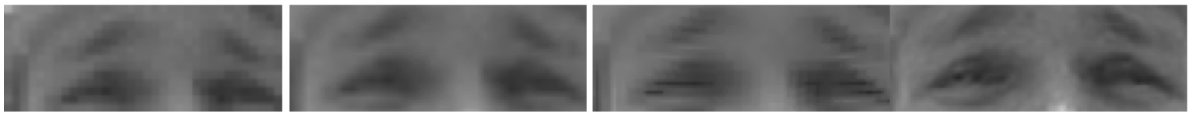

Figure 7. Detail of the 21th frame from "Foreman" sequence. From left to right: Low Quality, Linear Interpolation SRR with Probability Motion Estimation and 3-D ISKR.

function with negative input. In other words,

$$y = e^{-x} \tag{39}$$

Result of each frame is shown in the fourth column in Fig. 3. Generally speaking, the exponential f-functions give worse result than the simple linear functions with negative slope.

Last but not least, the Gaussian Function, which is also an intuitive choice of f-function, also performs reasonably well. In fact, it gives results that are comparable to the best result given by linear functions with negative slope. The type of Gaussian Function chosen was the probability density function for Gaussian distribution. The equation is shown as follows.

$$y = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{40}$$

In the project, since it is required that the f-function is monotonically non-increasing at positive x, μ, the mean of Normal equation is fixed at 0. Thus, the only parameter is , the standard deviation when the above function represents a Gaussian distribution.

The variable of PSNR with variation in σ is shown in Fig. 8.The best result are generally obtained around $\sigma = 1$.

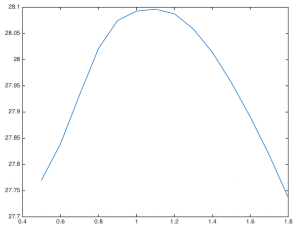

Figure 8. Gaussian Function, PSNR Vs σ

The best result obtained is PSNR 28.0966. Which is satisfactory considering its simplicity and ease of implementation. Although the PSNR value obtained with Gaussian functions are close to the best PSNR values given by linear function with negative slope, the variation of PSNR caused by varying parameter of Gaussian function is comparatively smaller than that of linear function with negative slope. In other words, its performance is comparatively stable. A more stable choice of f-function is less vulnerable to over train of parameter. Therefore, among all f-functions that have been experimented, Gaussian function is the best performing type.

PSNR with different SRR methods are shown in Tab.1.

SRR Method	PSNR
Linear Interpolation	27.9033
Linear Funtion	28.0494
Exponential Function	27.739
Gaussian Function	28.0966

Table 1. Results of the 8th frame from "Suzie" sequence.

4.2. SRR with 3-D Steering Kernel Regression

Fig. 6 shows the results of this algorithm and Fig. 7 shows the detial in the 21th frame.

Reconstruction image with 3-D ISKR is much more clear and has little disturbing artifacts than results from linear interpolation and SRR with Probablity Motion Estimation. Here SRR with Probability Motion Estimation is not as good as Linear Interpolation, and the reason is that the used "Foreman" Sequence has very large motion. Linear Interpolation does not take motion into consideration and Probability Motion Estimation is not good with large motion. In contrast, since the first step of 3-D ISKR is motion compensation, it is much better for video sequence with large mototion.

PSNR with different SRR methods are shown in Tab.2.

SRR Method	PSNR
Linear Interpolation	27.7493
Prob Motion Estimation	24.5002
3-D ISKR	32.76577

Table 2. Results of the 8th frame from "Foreman" sequence.

5. Conclusion

As discussed above, in this project, two SRR algorithms without explicit sub-pixel motion estimation were explored and implemented. Results show that, the SRR with Probabilistic Motion Estimation gives less clear image than 3-D ISKR algorithm. However, since 3-D ISKR is highly time-consuming, when the motions between frames are not very large, the SRR with Probabilistic Motion Estimation is a comparatively choice.

References

[1] A. Bouchachia. *Intelligence for nonlinear dynamics and synchronisation*, volume 3. Springer Science & Business Media, 2010.

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2001.

[3] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising with block-matching and 3d filtering. In *Electronic Imaging 2006*, pages 606414–606414. International Society for Optics and Photonics, 2006.

[4] A. Danielyan, A. Foi, V. Katkovnik, and K. Egiazarian. Image upsampling via spatially adaptive block-matching filtering. In *Signal Processing Conference, 2008 16th European*, pages 1–5. IEEE, 2008.

[5] M. Elad and D. Datsenko. Example-based regularization deployed to super-resolution reconstruction of a single image. *The Computer Journal*, 52(1):15–30, 2009.

[6] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 349–356. IEEE, 2009.

[7] M. Protter and M. Elad. Super resolution with probabilistic motion estimation. *Image Processing, IEEE Transactions on*, 18(8):1899–1904, 2009.

[8] M. Protter, M. Elad, H. Takeda, and P. Milanfar. Generalizing the nonlocal-means to super-resolution reconstruction. *Image Processing, IEEE Transactions on*, 18(1):36–51, 2009.

[9] H. Takeda, P. Milanfar, M. Protter, and M. Elad. Super-resolution without explicit subpixel motion estimation. *Image Processing, IEEE Transactions on*, 18(9):1958–1975, 2009.

[10] K. Zhang, X. Gao, D. Tao, and X. Li. Single image super-resolution with non-local means and steering kernel regression. *Image Processing, IEEE Transactions on*, 21(11):4544–4556, 2012.