A FRAMELET ALGORITHM FOR ENHANCING VIDEO STILLS

RAYMOND H. CHAN, ZUOWEI SHEN, AND TAO XIA

Abstract. High-resolution image reconstruction refers to the problem of constructing a high resolution image from low resolution images. One approach for the problem is the recent framelet method in [4]. There the low resolution images are assumed to be small perturbation of a reference image perturbed in different directions. Video clips are made of many still frames, usually about 30 frames per second. Thus most of the frames can be considered as small perturbations of their nearby frames. In particular, frames close to a specified reference frame can be considered as small perturbations of the reference frame. Hence the setting is similar to that in high-resolution image reconstruction. In this paper, we propose a framelet algorithm similar to that in [4] to enhance the resolution of any specified reference frames in video clips. Experiments on actual video clips show that our method can provide information that are not discernable from the given clips.

1. Introduction

High-resolution image reconstruction refers to the problem of constructing a high resolution image from low resolution images. Its applications include remote sensing, surveillance, and medical imaging. One of the methods to obtain the low resolution images is to use a sensor array with many low resolution sensors. In the array, each sensor is perturbed by subpixel displacements so as to provide enough independent information in the low resolution images to reconstruct the high resolution image, see [1]. The reconstruction problem can be viewed as a deconvolution problem where many methods are available. One approach is the recent wavelet algorithms developed in [3, 4].

In this paper, we will extend the method in [4] to video clips to enhance their resolution. Video clips are made of many still frames—normally 25 to 30 frames per second. Thus most of the frames can be considered as small perturbations of their nearby frames (nearby in time when the frames are captured). More precisely, consider a sequence of frames $\{f_k\}_{k=-K}^{K}$ in a given video clip, where k increases with the time when the frame f_k is captured. Let f_0 be the frame that we want to enhance its resolution. We will call f_0 the *reference frame* (or the *still*) and aim to improve its resolution by incorporating information from other frames in $\{f_k\}_{k=-K}^{K}$. For frames that are taken close to f_0 in time, they can be considered as small spatial perturbations of f_0. In other words, they can be considered as images obtained by displacing the sensors in a sensor array, and hence we have a setting similar to that of the high-resolution image reconstruction in [1]. This allows us to use the framelet algorithm developed in [4] to improve the resolution of f_0 by incorporating the information in $\{f_k\}_{k=-K}^{K}$.

Key words and phrases. Tight Frame System, Resolution Enhancement, Denoising.

R. Chan is with the Department of Mathematics, The Chinese University of Hong Kong, Shatin, Hong Kong, China. Email: rchan@math.cuhk.edu.hk. This work was supported by HKRGC Grant CUHK 400503 and CUHK DAG 2060257.

Z.W. Shen and T. Xia are with the Department of Mathematics, National University of Singapore, 2 Science Drive 2, Singapore 117543. Email: matzuows@nus.edu.sg, iesxiat@nus.edu.sg. Research supported in part by several grants at the National University of Singapore.

The model in [1] and the framelet method in [4] both assume that the perturbation of the sensors are translation only. However, in video clips, there are other motion effects from one frame to another. Therefore in this paper, we will develop ways to remove them in the frames and to estimate the translational displacement between frames f_k and the reference frame f_0. Once the displacement is determined, we can apply the method in [4]. One important difference between our method here and that in [4] is that the latter assumes that the low resolution images from all sensors are available. Here, the situation is more dynamic, and one has no way in knowing beforehand which sensor a frame f_k is corresponding to, and if we have enough frames to match all sensors. Thus our algorithm here is a dynamic version of that in [4]. For each frame coming into our algorithm, we will check if it is close enough to f_0 to be useful. If it is, then we compute its displacement and the sensor it is corresponding to. Then we apply the algorithm in [4] to it to enhance the resolution of f_0. This is done frame by frame. Unlike the setting in [1, 4] where a blurring operator for the problem is explicitly available, here we do not have any explicit blurring operator.

Our experimental results on two video clips (one filmed by us [5] and another provided to us by Ji and Fermuller [10]) show that our method can resolve textual details in the stills that are not discernable from the stills or the video clips themselves. Our method is therefore useful in revealing hidden information in video clips.

The outline of the paper is as follows. In §2, we give our tight frame algorithm in the 1-dimensional setting, i.e. for high-resolution *signal* reconstruction. Then in §3, we extend the algorithm to 2-dimensional case, i.e. high-resolution *image* reconstruction. The algorithm is different from that in [4], but will be more suitable in our video clips setting. In §4, we present our algorithm for enhancing video stills. For this, we need to remove the motion effects other than translations in the frames. Then we need to estimate the displacement errors between the frames. In §5, we illustrate the effectiveness of our algorithm by applying it on two video clips. Finally, conclusions are given in §6.

We first give the notations that we will be using. Bold-face characters indicate vectors, for instance, $\mathbf{g} = [\cdots, g(-1), g(0), g(1), \cdots]$. If f represents an image $f(x, y)$, then $\mathbf{f}$ represents the column vector of f constructed by raster scanning of f row by row. The matrix A^t denotes the transpose of A. Symbol I_j denotes the j-by-j identity matrix.

2. High-resolution Signal Reconstruction

To simplify the discussions, we start from the 1D signal reconstruction. The model is the 1D version of the model given in [1, 4].

2.1. **The Model.**

If we want to get a sampling of a continuous signal f, one conventional way is to take its integral, i.e.

$$f(n) \equiv \frac{1}{T} \int_{(n-\frac{1}{2})T}^{(n+\frac{1}{2})T} f(x)dx, \tag{2.1}$$

where T is the sampling length. Due to the limit of the capturing devices, sometimes we can only get the discrete signal at coarse sampling resolution, namely, TL, where L is a positive integer. The high-resolution signal reconstruction refers to the construction of a signal of sampling resolution T from a set of signals of sampling resolution LT. In this paper, for simplicity we consider only $L = 2$. This would imply that the resolution of our signals can only be enhanced by two times. However, larger value of L can be considered similarly, but with more complicated notations.

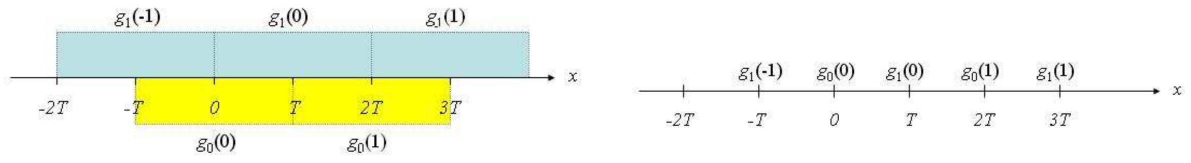

FIGURE 1. (Left) The domain of integration for $g_i(n)$ and (Right) the values of g.

Let the signals of sampling resolution $2T$ be positioned at nT, $n \in \mathbb{Z}$, and the shift between them is T, i.e.

$$g_i(n) = \frac{1}{2T}\int_{(2(n-\frac{1}{2})+i)T}^{(2(n+\frac{1}{2})+i)T} f(x)dx, \quad i = 0,1, \quad n \in \mathbb{Z}, \tag{2.2}$$

see Figure 1 (Left) to see the domain of integration for $g_i(n)$. A straightforward way to form a signal with sampling resolution T is to interlace the two low resolution signals g_0 and g_1, i.e.

$$g(n) = g_i(\lfloor \frac{n}{2} \rfloor), \qquad (i = n \mod 2),$$

see Figure 1 (Right). The function g is called the *observed high resolution signal.* In vector forms, we have $\mathbf{g} = \mathbf{g}_0 \otimes \mathbf{e}_0 + \mathbf{g}_1 \otimes \mathbf{e}_1$ where $\mathbf{e}_0 = (1,0)^t$, $\mathbf{e}_1 = (0,1)^t$, and $\otimes$ is the Kronecker product.

Define the sampling matrix D_i and the synthetic matrix U_i as

$$D_i = I_M \otimes \mathbf{e}_i^t \quad \text{and} \quad U_i = I_M \otimes \mathbf{e}_i, \qquad i = 0,1, \tag{2.3}$$

where M is the length of $\mathbf{g}_i$ (M can be infinite here). Then we also have

$$\mathbf{g}_i = D_i\mathbf{g} \quad \text{and} \quad \mathbf{g} = U_0\mathbf{g}_0 + U_1\mathbf{g}_1. \tag{2.4}$$

We note that the matrix U_i synthesizes $\mathbf{g}$ from the low-resolution signals $\mathbf{g}_i$, whereas D_i extracts $\mathbf{g}_i$ back from $\mathbf{g}$. Moreover, we have

$$U_0D_0 + U_1D_1 = I_N, \quad D_iU_i = I_M, \quad \text{and} \quad D_iU_j = 0 \quad \text{if} \quad i \neq j. \tag{2.5}$$

Here $N = 2M$ is the length of $\mathbf{g}$.

Though g is already a signal of sampling resolution T, and is supposedly better than the low resolution signals g_0 or g_1, it is not equal to the desired signal f given in (2.1). There is however a relationship between f and g that one can prove easily: if $f(x)$ is constant in the intervals $[(n-\frac{1}{2})T, (n+\frac{1}{2})T)$, $n \in \mathbb{Z}$, i.e. $f(x) = f(n)$, for $x \in [(n-\frac{1}{2})T, (n+\frac{1}{2})T)$, then

$$g(n) = \frac{1}{4}f(n-1) + \frac{1}{2}f(n) + \frac{1}{4}f(n+1), \quad \forall n \in \mathbb{Z}.$$

In matrix forms, it is

$$\mathbf{g} = \mathbf{H}\mathbf{f}, \tag{2.6}$$

where $\mathbf{H}$ is a Toeplitz matrix with entries $[H]_{ij} = a_{i-j}$, where

$$[\cdots, 0, a_{-1}, a_0, a_1, 0, \cdots] = [\cdots, 0, \frac{1}{4}, \frac{1}{2}, \frac{1}{4}, 0, \cdots]. \tag{2.7}$$

To obtain a better signal than $\mathbf{g}$, one will have to solve $\mathbf{f}$ from (2.6). It is an ill-posed inverse problem where many methods are available. One of them is the recent wavelet and framelet approaches developed in [3, 4].

2.2. **Tight Frame Approach.** The problem of high-resolution reconstruction is understood and analyzed under the framework of multiresolution analysis of $\mathcal{L}^2(\mathbb{R})$ by recognizing that the filter $\mathbf{a}$ given in (2.7) is a low-pass filter associated with a multi-resolution analysis [4]. More precisely, the following filters form tight frame filters by applying the unitary extension principle of [14, 4]:

$$\begin{aligned} m_0 &\equiv \frac{1}{2}[\frac{1}{2}, 1, \frac{1}{2}] = \mathbf{a}, \\ m_1 &\equiv \frac{\sqrt{2}}{4}[-1, 0, 1], \\ m_2 &\equiv \frac{1}{2}[-\frac{1}{2}, 1, -\frac{1}{2}]. \end{aligned} \tag{2.8}$$

In (2.6), the matrix $\mathbf{H}$ is of infinite size. In practice, $\mathbf{f} = [f_0, f_1, \cdots, f_{N-1}]$ will be a signal of finite length N. The corresponding finite matrix $\mathbf{H}$ will vary according to which boundary condition we use. The most popular boundary conditions are the Dirichlet, periodic, symmetric and the Neumann conditions. In this paper, we use the Neumann boundary condition as it usually gives better performance [12], and is required in the convergence proof in §2.4.

To derive our algorithm, we need one more notation. The Toeplitz-plus-Hankel matrix for a sequence $\mathbf{b} = [b_{-N}, b_{-N+1}, \dots, b_0, \dots, b_{N-1}, b_N]$ is defined by

$$ToeplitzHankel(\mathbf{b}) = [Hankel(\mathbf{b})_l + Toeplitz(\mathbf{b}) + Hankel(\mathbf{b})_r],$$

where

$$Hankel(\mathbf{b})_l = \begin{pmatrix} b_1 & \cdots & b_N \\ \vdots & \iddots & \\ b_N & & 0 \end{pmatrix}, \quad Hankel(\mathbf{b})_r = \begin{pmatrix} 0 & & b_{-N} \\ & \iddots & \vdots \\ b_{-N} & \cdots & b_{-1} \end{pmatrix},$$

and

$$Toeplitz(\mathbf{b}) = \begin{pmatrix} b_0 & b_{-1} & \cdots & b_{-N-2} & b_{-N-1} \\ b_1 & b_0 & \cdots & \cdots & b_{-N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b_{N-2} & \cdots & \cdots & b_0 & b_{-1} \\ b_{N-1} & b_{N-2} & \cdots & b_1 & b_0 \end{pmatrix}.$$

Note that under the Neumann boundary condition, the matrix for a filter $\mathbf{b}$ will be the matrix given by $ToeplitzHankel(\mathbf{b})$, see [12].

Under the Neumann boundary condition, the matrix form of the forward and inverse tight frame transforms can be represented by the matrices $\{\mathbf{T}_p\}_{p=0}^2$ and $\left\{\mathbf{T}_p^*\right\}_{p=0}^2$ defined by

$$\mathbf{T}_p = ToeplitzHankel(\mathbf{a}_p) \quad \text{and} \quad \mathbf{T}_p^* = ToeplitzHankel(\bar{\mathbf{a}}_p). \tag{2.9}$$

Here $\mathbf{a}_p$ and $\bar{\mathbf{a}}_p$ are $(2N+1)$-vectors given by:

$$\begin{aligned} \mathbf{a}_p &= [0, \cdots, 0, m_p(-1), m_p(0), m_p(1), 0, \cdots, 0], \\ \bar{\mathbf{a}}_p &= [0, \cdots, 0, m_p(1), m_p(0), m_p(-1), 0, \cdots, 0], \end{aligned}$$

with m_p given in (2.8). The perfect reconstruction formula for framlet transform is $I = \sum_{i=0}^2 \mathbf{T}_i^* \mathbf{T}_i$.

Since the filter $\mathbf{a}$ in (2.7) equals m_0, we have $\mathbf{H} = \mathbf{T}_0$. Therefore, by (2.6), we have

$$\mathbf{f} = \sum_{i=0}^{2} \mathbf{T}_i^* \mathbf{T}_i \mathbf{f} = \mathbf{T}_0^* \mathbf{g} + \mathbf{T}_1^* \mathbf{T}_1 \mathbf{f} + \mathbf{T}_2^* \mathbf{T}_2 \mathbf{f}. \tag{2.10}$$

This is the equation from which we solve for $\mathbf{f}$ in the following. Before doing so, we will like to add to the equation one more complication from the real world.

In actual applications, especially when enhancing videos, (2.2) does not hold in general. In fact, it is almost impossible to require an object that appears in the reference frame to be moved by exactly half subpixel length in any other frame. Rather in any one frame, it may move to a distance of ℓ pixels away where $\ell = n + \epsilon$, with $n \in \mathbb{Z}$ and $|\epsilon| \leq 1/2$. In that case, we can shift the frame back by n pixels and consider the shifted frame as a displaced frame of the reference frame with displacement equals ϵ. More precisely, instead of having the perfectly aligned low resolution signals $g_i(\cdot)$ at half subpixel length as in Figure 1 (Left), what we obtained are the shifted versions of g_i, i.e. we have

$$\widetilde{g}_i(\cdot) \equiv g_i(\cdot + \epsilon_i), \quad 0 \leq |\epsilon_i| < \frac{1}{2}, \quad i = 0, 1. \tag{2.11}$$

(If $|\epsilon_i| = 1/2$, then the frame is moved exactly half subpixel, and hence is a frame obtained by the other sensor.) The parameters ϵ_i are called the *displacement errors*. As in (2.4) and (2.6), $\tilde{g}_i$ can be considered as the down-sample of f after it has passed through a filter of the form $\frac{1}{2}[\frac{1}{2} - \epsilon_i, 1, \frac{1}{2} + \epsilon_i]$. More precisely, we have

$$\tilde{\mathbf{g}}_i = D_i \mathbf{H}(\epsilon_i)\mathbf{f}, \quad i = 0, 1, \tag{2.12}$$

where D_i is defined in (2.3), $\mathbf{H}(\epsilon) = Toeplitz(\mathbf{a}(\epsilon))$, and $\mathbf{a}(\epsilon) = \frac{1}{2}[\frac{1}{2} - \epsilon, 1, \frac{1}{2} + \epsilon]$.

One may verify that $\mathbf{a}(\epsilon) = m_0 + \sqrt{2}\epsilon \cdot m_1$, with m_0 and m_1 defined in (2.8), see [4]. Therefore

$$\mathbf{H}(\epsilon) = \mathbf{H} + \sqrt{2}\epsilon \mathbf{T}_1,$$

and hence

$$\mathbf{H}\mathbf{f} = \mathbf{H}(\epsilon)\mathbf{f} - \sqrt{2}\epsilon \mathbf{T}_1 \mathbf{f}.$$

In particular, $\mathbf{g}_i$ can be obtained from $\tilde{\mathbf{g}}_i$ via

$$\mathbf{g}_i = D_i \mathbf{H}\mathbf{f} = \tilde{\mathbf{g}}_i - \sqrt{2} D_i \epsilon_i \mathbf{T}_1 \mathbf{f}, \qquad i = 0, 1. \tag{2.13}$$

Thus by (2.13), the low resolution signal $\mathbf{g}_i$ satisfying (2.6) can be extracted from the shifted signal $\tilde{\mathbf{g}}_i$ by correcting the displacement errors.

Define $\widetilde{\mathbf{g}} = U_0 \widetilde{\mathbf{g}}_0 + U_1 \widetilde{\mathbf{g}}_1$ and

$$\mathbf{g}_\epsilon = \sqrt{2}(U_0 D_0 \epsilon_0 + U_1 D_1 \epsilon_1)\mathbf{T}_1 \mathbf{f}. \tag{2.14}$$

Then by (2.5),

$$\mathbf{g} = \mathbf{T}_0 \mathbf{f} = \mathbf{H}\mathbf{f} = (U_0 D_0 + U_1 D_1)\mathbf{H}\mathbf{f} = \widetilde{\mathbf{g}} - \mathbf{g}_\epsilon.$$

Substituting this into (2.10), we finally have the high-resolution reconstruction equation for signals with displacement errors:

$$\mathbf{f} = \mathbf{T}_0^*(\widetilde{\mathbf{g}} - \mathbf{g}_\epsilon) + \sum_{i=1}^{2} \mathbf{T}_i^* \mathbf{T}_i \mathbf{f}. \tag{2.15}$$

Our algorithm is basically to iterate on this equation and (2.14). This is done in the next section.

2.3. **Resolution Enhancing Algorithm.** Signals and images are usually contaminated with noise, which are of high-frequency in nature. Equation (2.15) makes it easy to remove them. In fact, since $\mathbf{T}_1$ and $\mathbf{T}_2$ are high-pass filters, noise are emphasized in the second term in (2.15). Hence a thresholding approach can be used to denoise the signal or the image. Here we use a framelet thresholding approach in [2].

For a given signal $\mathbf{f}$, the first step in the denoising procedure is to decompose $\mathbf{f}$ in different levels in the framelet domain so as to decorrelate the signal. For this, we need to define the framelet decomposition operator $\mathcal{A}_J$ and its adjoint reconstruction operator $\mathcal{A}_J^*$. Here J is the number of levels we want to decompose our signals. To write the decomposition and reconstruction in convolution forms, the filters used in the decomposition above the 0th level need to be dilated. For levels $0 < j \le J$, the dilated filters, denoted by $\{m_{l,j}\}_{l=0}^2$, are defined by

$$m_{l,j}(k) = \begin{cases} m_l(2^{-j+1}k), & k \in 2^{j-1}\mathbb{Z}, \\ 0, & k \in \mathbb{Z} \setminus 2^{j-1}\mathbb{Z}, \end{cases}$$

where m_l are given in (2.8). The corresponding Toeplitz matrix is $\mathbf{M}_{l,j} = Toeplitz(m_{l,j})$. With these notations, we have

$$\mathcal{A}_J = \begin{bmatrix} \prod_{j=J}^1 \mathbf{M}_{0,j} \\ \mathbf{M}_{1,J} \prod_{j=J-1}^1 \mathbf{M}_{0,j} \\ \mathbf{M}_{2,J} \prod_{j=J-1}^1 \mathbf{M}_{0,j} \\ \mathbf{M}_{1,J-1} \prod_{j=J-2}^1 \mathbf{M}_{0,j} \\ \vdots \\ \mathbf{M}_{1,1} \\ \mathbf{M}_{2,1} \end{bmatrix}$$

and

$$\mathcal{A}_J^* = [\prod_{j=1}^{J} \mathbf{M}_{0,j}^*, (\prod_{j=1}^{J-1} \mathbf{M}_{0,j}^*)\mathbf{M}_{1,J}^*, (\prod_{j=1}^{J-1} \mathbf{M}_{0,j}^*)\mathbf{M}_{2,J}^*, (\prod_{j=1}^{J-2} \mathbf{M}_{0,j}^*)\mathbf{M}_{1,J-1}^*, \cdots, \mathbf{M}_{1,1}^*, \mathbf{M}_{2,1}^*].$$

Once a signal has been decomposed by $\mathcal{A}_J$, the second step in the denoising procedure is to threshold the framelet coefficients for levels $j > 0$. For any N-vector $\mathbf{u}$, define the threshold operator $\mathcal{D}_\lambda$ as

$$\mathcal{D}_\lambda(\mathbf{u}) \equiv [t_\lambda(u(1)), t_\lambda(u(2)), \cdots, t_\lambda(u(N))]^t$$

where $t_\lambda(x) \equiv \operatorname{sgn}(x)\max(|x| - \lambda/2, 0)$ is the soft threshold function. The denoising operator $\mathcal{T}$, with thresholding parameters $\lambda_{i,j}, i = 1, 2,\ j = 1, \cdots, J$, is then defined as

$$\mathcal{T}\mathcal{A}_J\mathbf{v} = \begin{bmatrix} \prod_{j=J}^1 \mathbf{M}_{0,j}\mathbf{v} \\ \mathcal{D}_{\lambda_{1,J}}(\mathbf{M}_{1,J} \prod_{j=J-1}^1 \mathbf{M}_{0,j}\mathbf{v}) \\ \mathcal{D}_{\lambda_{2,J}}(\mathbf{M}_{2,J} \prod_{j=J-1}^1 \mathbf{M}_{0,j}\mathbf{v}) \\ \mathcal{D}_{\lambda_{1,J-1}}(\mathbf{M}_{1,J-1} \prod_{j=J-2}^1 \mathbf{M}_{0,j}\mathbf{v}) \\ \vdots \\ \mathcal{D}_{\lambda_{1,1}}(\mathbf{M}_{1,1}\mathbf{v}) \\ \mathcal{D}_{\lambda_{2,1}}(\mathbf{M}_{2,1}\mathbf{v}) \end{bmatrix}. \tag{2.16}$$

A typical choice for $\lambda_{i,j}$ is $\lambda_{i,j} = 2\|m_0\|^{j-1}\|m_i\|\sigma\sqrt{\log N}$, where $\|\cdot\|$ is the l_2 norm and σ is the variance of the Gaussian noise in the signal $\mathbf{v}$ estimated numerically by the method given in [7]. We note that for i.i.d. Guassian random variables $\{\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_j\}$ each with variance σ, if

$\varepsilon_N = \sum_{i=0}^{j} h_i\varepsilon_i$, then $var(\varepsilon_N) = \sigma\|[h_1, h_2, \cdots, h_j]\|$. Hence we have included the l_2 norms of the filters in $\lambda_{i,j}$ above.

Incorporating this denoising operator into (2.15), we get the equation for the resolution enhancement:

$$\mathbf{f} = \mathbf{T}_0^*(\widetilde{\mathbf{g}} - \mathbf{g}_\epsilon) + \sum_{i=1}^{2} \mathbf{T}_i^* \mathcal{A}_J^* \mathcal{T} \mathcal{A}_J \mathbf{T}_i \mathbf{f}. \tag{2.17}$$

Our algorithm is to iterate $\mathbf{f}$ in this equation:

$$\mathbf{f}_{n+1} = \mathbf{T}_0^*(\widetilde{\mathbf{g}} - \mathbf{g}_\epsilon) + \sum_{i=1}^{2} \mathbf{T}_i^* \mathcal{A}_J^* \mathcal{T} \mathcal{A}_J \mathbf{T}_i \mathbf{f}_n, \quad n = 0, 1, \cdots. \tag{2.18}$$

In the next subsection, we show that if $\mathbf{g}_\epsilon$ is fixed independent of $\mathbf{f}$, then (2.18) is convergent. However, from (2.14), $\mathbf{g}_\epsilon$ is dependent on $\mathbf{f}$. To overcome this, we use an alternate direction approach: we fix $\mathbf{g}_\epsilon$ and we iterate $\mathbf{f}$ by (2.18) until it converges. Once we get an updated $\mathbf{f}$, we compute an updated $\mathbf{g}_\epsilon$ using (2.14). This results in the following algorithm:

Algorithm 1. *Resolution Enhancement Algorithm for 1D Signal*

(1) *Let* $\hat{\mathbf{f}}^{(0)} = \mathbf{T}_0^*\widetilde{\mathbf{g}}$ *and* $m = 1$.

(2) *Let* $\mathbf{f}_0^{(m)} = \hat{\mathbf{f}}^{(m-1)}$ *and* $\mathbf{g}_\epsilon^{(m)} = \sum_{i=0}^{1} \sqrt{2} U_i D_i \epsilon_i \mathbf{T}_1 \hat{\mathbf{f}}^{(m-1)}$.

(3) *Iterate on* $\mathbf{f}_n^{(m)}$ *until it converges:*

$$\mathbf{f}_{n+1}^{(m)} = \mathbf{T}_0^*(\widetilde{\mathbf{g}} - \mathbf{g}_\epsilon^{(m)}) + \sum_{i=1}^{2} \mathbf{T}_i^* \mathcal{A}_J^* \mathcal{T} \mathcal{A}_J \mathbf{T}_i \mathbf{f}_n^{(m)}, \quad n = 0, 1, \cdots.$$

(4) *Set* $\hat{\mathbf{f}}^{(m)} = \mathbf{f}_n^{(m)}$ *when converge.*

(5) *If* $\|\hat{\mathbf{f}}^{(m)} - \hat{\mathbf{f}}^{(m-1)}\| > tol$*, then set* $m + 1 \to m$ *and go to Step (2). Otherwise, end.*

Numerical tests on real data show that both $\mathbf{f}$ and $\mathbf{g}_\epsilon$ converges within 3–6 iterations, see the numerical results in §5.

2.4. Convergence of the Algorithm.

In this section, we prove the convergence of (2.18). The following lemma, given in [6, Lemma 2.2], is needed for the proof.

Lemma 2.1. *The thresholding operator* $\mathcal{D}_\lambda$ *is non-expansive, i.e. for any two vectors* $\mathbf{v}_1$ *and* $\mathbf{v}_2$,

$$\|\mathcal{D}_\lambda(\mathbf{v}_1) - \mathcal{D}_\lambda(\mathbf{v}_2)\| \le \|\mathbf{v}_1 - \mathbf{v}_2\|.$$

Furthermore, if $\mathcal{T}$ *is the denoising operator defined by (2.16), then*

$$\|\mathcal{T}\mathcal{A}_J(\mathbf{v}_1) - \mathcal{T}\mathcal{A}_J(\mathbf{v}_2)\| \le \|\mathbf{v}_1 - \mathbf{v}_2\|.$$

In particular, $\mathcal{T}\mathcal{A}_J$ *is continuous and* $\|\mathcal{T}\mathcal{A}_J(\mathbf{v})\| \le \|\mathbf{v}\|$ *for all vectors* $\mathbf{v}$.

We note also that the N-by-N matrix $\mathbf{T}_0 = ToeplitzHankel(m_0)$ can be diagonalized and its minimum eignvalue is equal to $\cos^2(\frac{N-1}{2N}\pi) > 0$, see [12]. In other words, $\mathbf{T}_0$ is nonsingular.

Theorem 2.2. *If* $\mathbf{T}_0$ *is nonsingular and* $\mathbf{g}_\epsilon$ *is fixed, then (2.18) converges for any* $\mathbf{f}_0$.

Proof. Because $\mathbf{T}_0$ is nonsingular and $I = \mathbf{T}_0^*\mathbf{T}_0 + \sum_{i=1}^2 \mathbf{T}_i^*\mathbf{T}_i$, there exists a constant $\mu < 1$, such that $\left\|\sum_{i=1}^2 \mathbf{T}_i^*\mathbf{T}_i\right\| = \|I - \mathbf{T}_0^*\mathbf{T}_0\| \le \mu$. Let $\mathbf{T}^* \equiv [\mathbf{T}_1^*, \mathbf{T}_2^*]$. Then

$$\|\mathbf{T}\|^2 = \max_{\|\mathbf{u}\|=1} \|\mathbf{T}\mathbf{u}\|^2 = \max_{\|\mathbf{u}\|=1} \mathbf{u}^* \sum_{i=1}^2 \mathbf{T}_i^*\mathbf{T}_i\mathbf{u} = \left\|\sum_{i=1}^2 \mathbf{T}_i^*\mathbf{T}_i\right\| \le \mu.$$

For any given $\mathbf{f}_0$ and any positive integers k and n, according to (2.18), we have

$$\begin{aligned}
\|\mathbf{f}_{n+k} - \mathbf{f}_n\| &= \left\|\sum_{i=1}^2 \mathbf{T}_i^*\mathcal{A}_J^*(\mathcal{T}\mathcal{A}_J\mathbf{T}_i\mathbf{f}_{n+k-1} - \mathcal{T}\mathcal{A}_J\mathbf{T}_i\mathbf{f}_{n-1})\right\| \\
&= \left\|\mathbf{T}^*\mathcal{A}_J^*\begin{bmatrix} \mathcal{T}\mathcal{A}_J\mathbf{T}_1\mathbf{f}_{n+k-1} - \mathcal{T}\mathcal{A}_J\mathbf{T}_1\mathbf{f}_{n-1} \\ \mathcal{T}\mathcal{A}_J\mathbf{T}_2\mathbf{f}_{n+k-1} - \mathcal{T}\mathcal{A}_J\mathbf{T}_2\mathbf{f}_{n-1} \end{bmatrix}\right\| \\
&\le \|\mathbf{T}^*\| \left\|\begin{bmatrix} \mathcal{T}\mathcal{A}_J\mathbf{T}_1\mathbf{f}_{n+k-1} - \mathcal{T}\mathcal{A}_J\mathbf{T}_1\mathbf{f}_{n-1} \\ \mathcal{T}\mathcal{A}_J\mathbf{T}_2\mathbf{f}_{n+k-1} - \mathcal{T}\mathcal{A}_J\mathbf{T}_2\mathbf{f}_{n-1} \end{bmatrix}\right\| \\
&\le \|\mathbf{T}^*\| \left\|\begin{bmatrix} \mathbf{T}_1\mathbf{f}_{n+k-1} - \mathbf{T}_1\mathbf{f}_{n-1} \\ \mathbf{T}_2\mathbf{f}_{n+k-1} - \mathbf{T}_2\mathbf{f}_{n-1} \end{bmatrix}\right\| \\
&\le \|\mathbf{T}^*\| \, \|\mathbf{T}\| \, \|\mathbf{f}_{n+k-1} - \mathbf{f}_{n-1}\| \\
&\le \mu \|\mathbf{f}_{n+k-1} - \mathbf{f}_{n-1}\| \le \mu^n \|\mathbf{f}_k - \mathbf{f}_0\|.
\end{aligned}$$

Using similar arguments on (2.18), we also get

$$\|\mathbf{f}_n\| \le \|\widetilde{\mathbf{g}} - \mathbf{g}_\epsilon\| + \mu \|\mathbf{f}_{n-1}\|.$$

Applying this recursively from n back to 0, we then get

$$\|\mathbf{f}_n\| \le \frac{1}{1-\mu} \|\widetilde{\mathbf{g}} - \mathbf{g}_\epsilon\| + \|\mathbf{f}_0\|,$$

i.e., $\|\mathbf{f}_n\|$ is bounded. Thus the sequence $\{\mathbf{f}_n\}_n$ is a Cauchy sequence and the limit exists. □

3. High-resolution Image Reconstruction

Next we extend the discussion of the previous section to 2D case.

3.1. The Model.

3.1. The Model. High-resolution image reconstruction refers to the problem of constructing an image with resolution N-by-N using low-resolution images of resolution M-by-M. For our discussion here, it suffices to consider $N = 2M$. Similar to the 1D case in the last section, the desired high resolution image is modeled as

$$f(n_1, n_2) \equiv \frac{1}{T^2} \int_{(n_2-\frac{1}{2})T}^{(n_2+\frac{1}{2})T} \int_{(n_1-\frac{1}{2})T}^{(n_1+\frac{1}{2})T} f(x,y)dxdy,$$

whereas we are given the low resolution images $g_{i,j}$:

$$g_{i,j}(n_1, n_2) = \frac{1}{4T^2} \int_{(2(n_2-\frac{1}{2})+i)T}^{(2(n_2+\frac{1}{2})+i)T} \int_{(2(n_1-\frac{1}{2})+j)T}^{(2(n_1+\frac{1}{2})+j)T} f(x,y)dxdy, \quad 0 \le i,j < 2, \quad n_1, n_2 \in \mathbb{Z}.$$

Here each $g_{i,j}$ differs from the other by a displacement of length T.

The observed high resolution image g can be composed in the following way

$$g_{i,j}[n_1, n_2] = g[2n_1 + i, 2n_2 + j], \quad 0 \le i,j < 2, \quad n_1, n_2 \in \mathbb{Z}. \tag{3.1}$$

We note that each low resolution image $g_{i,j}$ is the downsampled image of the observed image g at specified sensor position (i,j). Therefore we will call $g_{i,j}$ the low resolution image at downsampling position (i,j), or in short the (i,j)th image. The question again is to find f from g, see [1].

Define the 2D downsampling and upsampling matrices $D_{i,j} = D_j \otimes D_i$ and $U_{i,j} = U_j \otimes U_i$. Then we have, $\mathbf{g}_{i,j} = D_{i,j}\mathbf{g}$ and $\mathbf{g} = \sum_{i,j=0}^{1} U_{i,j}\mathbf{g}_{i,j}$. Again, $U_{i,j}$ synthesizes $\mathbf{g}$ from the low resolution images $\mathbf{g}_{i,j}$, whereas $D_{i,j}$ extracts the image $\mathbf{g}_{i,j}$ back from $\mathbf{g}$. We note that

$$\sum_{i,j=0}^{1} U_{i,j}D_{i,j} = I_{N^2}, \quad D_{i,j}U_{i,j} = I_{M^2}, \quad \text{and} \quad D_{i_1,j_1}U_{i_2,j_2} = 0 \quad \text{if} \quad i_1 \neq i_2,\ j_1 \neq j_2.$$

The high-resolution image reconstruction can be modeled by (2.6) too with the filter matrix $\mathbf{H}$ in this case becomes $\mathbf{H} = \mathbf{T}_{0,0}$, where $\mathbf{T}_{0,0}$ is the low pass filter corresponding to the 2D framelet transform which is the tensor product of its 1D cousin. More precisely, the 2D framelet filter matrices are $\mathbf{T}_{p,q} = \mathbf{T}_q \otimes \mathbf{T}_p$ and $\mathbf{T}^*_{p,q} = \mathbf{T}^*_q \otimes \mathbf{T}^*_p$, where $\mathbf{T}_p$ and $\mathbf{T}^*_q$ are given in (2.9). We also have the perfect reconstruction formula:

$$\sum_{i,j=0}^{2} \mathbf{T}^*_{i,j}\mathbf{T}_{i,j} = I_{N^2}.$$

Multiplying by the high resolution image $\mathbf{f}$ that we are seeking on both sides and substituting $\mathbf{T}_{0,0}\mathbf{f} = \mathbf{H}\mathbf{f}$ by $\mathbf{g}$, we have

$$\mathbf{f} = \mathbf{T}^*_{0,0}\mathbf{g} + \sum_{\substack{i,j=0 \\ (i,j)\neq(0,0)}}^{2} \mathbf{T}^*_{i,j}\mathbf{T}_{i,j}\mathbf{f}. \tag{3.2}$$

3.2. **Correcting Displacement Errors.** In practice, the low resolution images may not be aligned exactly by length T as in (3.1) and displacement errors may exist. Similar to the 1D case, we can correct the displacement error in each low resolution image by framelet systems.

Let the obtained low resolution images be

$$\widetilde{g}_{i,j}(\cdot,\cdot) = g_{i,j}(\cdot + \epsilon^x_{i,j}, \cdot + \epsilon^y_{i,j}) \tag{3.3}$$

rather than $g_{i,j}(\cdot,\cdot)$, where

$$0 \leq |\epsilon^x_{i,j}|, |\epsilon^y_{i,j}| < \frac{1}{2}, \quad i,j = 0,1, \tag{3.4}$$

see (2.11). Similar to (2.13), we have

$$\mathbf{g}_{i,j} = \widetilde{\mathbf{g}}_{i,j} - D_{i,j}\left(\sqrt{2}\epsilon^x_{i,j}\mathbf{T}_{0,1} + \sqrt{2}\epsilon^y_{i,j}\mathbf{T}_{1,0} + 2\epsilon^x_{i,j}\epsilon^y_{i,j}\mathbf{T}_{1,1}\right)\mathbf{f},$$

Let $\mathbf{g}_\epsilon = \sum_{i,j=0}^{1} U_{i,j}D_{i,j}\left(\sqrt{2}\epsilon^x_{i,j}\mathbf{T}_{0,1} + \sqrt{2}\epsilon^y_{i,j}\mathbf{T}_{1,0} + 2\epsilon^x_{i,j}\epsilon^y_{i,j}\mathbf{T}_{1,1}\right)\mathbf{f}$ and $\widetilde{\mathbf{g}} = \sum_{i,j=0}^{1} U_{i,j}\widetilde{\mathbf{g}}_{i,j}$. Then

$$\mathbf{g} = \mathbf{T}_{0,0}\mathbf{f} = \mathbf{H}\mathbf{f} = \sum_{i,j=0}^{1} U_{i,j}D_{i,j}\mathbf{H}\mathbf{f} = \widetilde{\mathbf{g}} - \mathbf{g}_\epsilon. \tag{3.5}$$

Substituting this into (3.2), we have the high-resolution reconstruction equation for images with displacement errors:

$$\mathbf{f} = \mathbf{T}_{0,0}^*(\widetilde{\mathbf{g}} - \mathbf{g}_\epsilon) + \sum_{\substack{i,j=0 \\ (i,j)\neq(0,0)}}^{2} \mathbf{T}_{i,j}^*\mathbf{T}_{i,j}\mathbf{f}. \tag{3.6}$$

3.3. **The Algorithm.** As in §2.3, by embedding the denoising operator $\mathcal{A}_J$ into (3.6), we arrive at the 2D resolution enhancement formula:

$$\mathbf{f} = \mathbf{T}_{0,0}^*(\widetilde{\mathbf{g}} - \mathbf{g}_\epsilon) + \sum_{\substack{i,j=0 \\ (i,j)\neq(0,0)}}^{2} \mathbf{T}_{i,j}^*\mathcal{A}_J^*\mathcal{T}\mathcal{A}_J\mathbf{T}_{i,j}\mathbf{f}.$$

This leads to the following algorithm:

Algorithm 2. *Resolution Enhancement Algorithm for 2D Images*

(1) $\hat{\mathbf{f}}^{(0)} = \mathbf{T}_{0,0}^*\widetilde{\mathbf{g}}$ *and* $m = 1$.

(2) *Let* $\mathbf{f}_0^{(m)} = \hat{\mathbf{f}}^{(m-1)}$ *and*

$$\mathbf{g}_\epsilon^{(m)} = \sum_{i,j=0}^{1} U_{i,j}D_{i,j}\left(\sqrt{2}\epsilon_{i,j}^x\mathbf{T}_{0,1} + \sqrt{2}\epsilon_{i,j}^y\mathbf{T}_{1,0} + 2\epsilon_{i,j}^x\epsilon_{i,j}^y\mathbf{T}_{1,1}\right)\hat{\mathbf{f}}^{(m-1)}.$$

(3) *Iterate on* $\mathbf{f}_n^{(m)}$ *until it converges:*

$$\mathbf{f}_{n+1}^{(m)} = \mathbf{T}_{0,0}^*(\widetilde{\mathbf{g}} - \mathbf{g}_\epsilon^{(m)}) + \sum_{\substack{i,j=0 \\ (i,j)\neq(0,0)}}^{2} \mathbf{T}_{i,j}^*\mathcal{A}_J^*\mathcal{T}\mathcal{A}_J\mathbf{T}_{i,j}\mathbf{f}_n^{(m)}, \quad n = 0, 1, \cdots.$$

(4) *Set* $\hat{\mathbf{f}}^{(m)} = \mathbf{f}_n^{(m)}$ *when converge.*

(5) *If* $\left\|\hat{\mathbf{f}}^{(m)} - \hat{\mathbf{f}}^{(m-1)}\right\| > tol$, *then set* $m + 1 \to m$ *and go to Step (2). Otherwise, end.*

Remark 3.1. The convergence of Step (3) can be proven using similar arguments as in Theorem 2.2.

4. Resolution Enhancement for Video Clips

Video clips consist of many still frames. Each frame can be considered as perturbations of its nearby frames. Therefore we may generate images of higher resolution by exploiting the high redundancy between the nearby frames. In this paper, for simplicity, we consider only doubling the resolution of the reference frames. We will use the framelet method in §3. However, there are two problems in video still enhancement that do not appear in the image enhancement discussed in §3:

(1) For each frame, we have to estimate its sensor position and displacement errors with respect to the desired high-resolution reference frame.
(2) Not all low resolution images at all sensor positions are available.

In following subsections we will tackle these two questions. We will need to rectify Algorithm 2 to suit the video applications.

Consider a sequence of frames $\{f_k\}_{k=-K}^{K}$ in a given video clip, where k increases with the time when the frame f_k is captured. We aim to improve the resolution of the reference frame f_0 by incorporating information from frames $\{f_k\}_{k\neq 0}$. For simplicity, we may consider f_0 as the $(0,0)$th image without displacement error. In order to use the framelet method in §3, for each f_k, $k \neq 0$, we have to estimate the sensor position (i, j) it corresponds to and its displacement error.

The images $g_{i,j}$ and g in (3.1) can be considered as the discretized version of the underlying high-resolution image f at different sampling rate (see Figure 1 for 1D case). If we consider f being defined on the real domain, then extending the definition of $g_{i,j}$ and g on the real domain, we have the following half-pixel relationship between the low resolution images,

$$g_{i,j}(x, y) = g[2x + i, 2y + j] = g_{0,0}(x + \frac{i}{2}, y + \frac{j}{2}), \quad 0 \le i, j < 2,\ x, y \in \mathbb{R}.$$

If there are displacement errors in the low resolution images $\widetilde{g}_{i,j}(x, y)$, then (3.3) gives

$$\widetilde{g}_{i,j}(x, y) = g_{0,0}(x + \frac{i}{2} + \frac{\epsilon^x_{i,j}}{2}, y + \frac{j}{2} + \frac{\epsilon^y_{i,j}}{2}), \quad 0 \le i, j < 2,\ x, y \in \mathbb{R}.$$

This equation can be used to estimate the sensor position (s^x_k, s^y_k) and the displacement errors $(\epsilon^x_k, \epsilon^y_k)$ for the frame f_k with respect to f_0, i.e.

$$(s^x_k, s^y_k, \epsilon^x_k, \epsilon^y_k) = \arg\min_{\{0 \le i,j \le 1,\ 0 \le |\epsilon^x|, |\epsilon^y| < \frac{1}{2}\}} \sum_{x,y} [f_k(x - \frac{i}{2} - \frac{\epsilon^x}{2}, y - \frac{j}{2} - \frac{\epsilon^y}{2}) - f_0(x, y)]^2. \tag{4.1}$$

Equation (4.1) requires f_k to be a translation of f_0 only, which is generally not true in real applications. Therefore we first have to remove other motion effects in f_k before we can estimate the position and the displacement errors.

4.1. **Estimating the Motion Parameters.**

Rather than using displacement vector field as in [16], for computational efficiency, we restrict ourselves to affine transforms only, see [9]. In particular, we assume that the frames $\{f_k\}_{k \neq 0}$ are related to f_0 by a coordinate transform, i.e.

$$f_k(R_k\mathbf{x} - \mathbf{r}_k) \approx f_0(\mathbf{x}), \quad k \neq 0, \tag{4.2}$$

where $\mathbf{x}$ are the coordinates of the pixels in the region of interest, which may be the entire image or part of the image. Denote

$$\widetilde{\mathbf{x}} \equiv R\mathbf{x} - \mathbf{r} \equiv \begin{bmatrix} c_0 & c_1 \\ c_3 & c_4 \end{bmatrix} \mathbf{x} + \begin{bmatrix} c_2 \\ c_5 \end{bmatrix} = \begin{bmatrix} c_0 & c_1 & c_2 \\ c_3 & c_4 & c_5 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}. \tag{4.3}$$

Our task is to estimate the parameters $\{c_i\}_{i=0}^5$ for each frame $f \in \{f_k\}_{k=1}^K$. This is done by minimizing the sum of squares of the intensity between f and f_0 as mentioned in (4.1):

$$E(f, f_0) = \sum_{j \in \mathcal{I}} [f(R\mathbf{x}_j - \mathbf{r}) - f_0(\mathbf{x}_j)]^2 \equiv \sum_{j \in \mathcal{I}} e_j^2, \tag{4.4}$$

where $\mathcal{I}$ is the index set of pixels in the region of interest. Here and in the following, whenever $R\mathbf{x} - \mathbf{r} \notin \mathbb{Z}^2$, we will evaluate $f(R\mathbf{x} - \mathbf{r})$ by using the bilinear interpolation [13, pp. 126–132].

We solve (4.4) by using the Levenberg-Marquardt iterative nonlinear minimization algorithm [15]. It updates $\mathbf{c} = [c_0, \ldots, c_5]$ by an amount $\Delta\mathbf{c} = (A + \beta I)^{-1}\mathbf{b}$ where β is a time-varying stabilization parameter, and the entries of A and $\mathbf{b}$ are given by

$$a_{kl} = \sum_{j \in \mathcal{I}} \frac{\partial e_j}{\partial c_k} \frac{\partial e_j}{\partial c_l}, \quad \text{and} \quad b_k = -\sum_{j \in \mathcal{I}} e_j \frac{\partial e_j}{\partial c_k}, \quad 0 \le k, l \le 5. \tag{4.5}$$

The partial derivatives can be shown to be:

$$\frac{\partial e_j}{\partial c_0} = x_j \frac{\partial f}{\partial \tilde{x}}, \quad \frac{\partial e_j}{\partial c_1} = y_j \frac{\partial f}{\partial \tilde{x}}, \quad \frac{\partial e_j}{\partial c_2} = \frac{\partial f}{\partial \tilde{x}}, \quad \frac{\partial e_j}{\partial c_3} = x_j \frac{\partial f}{\partial \tilde{y}}, \quad \frac{\partial e_j}{\partial c_4} = y_j \frac{\partial f}{\partial \tilde{y}}, \quad \frac{\partial e_j}{\partial c_5} = \frac{\partial f}{\partial \tilde{y}}. \tag{4.6}$$

The advantage of using Levenberg-Marquardt over straightforward gradient descent is that it converges in fewer iterations [13, pp. 686–694].

This is the algorithm.

Algorithm 3. $(R, \mathbf{r}) \leftarrow$ *Affine*(f, f_0): *Estimating the parameters of the affine transform.*

```
(1) Initialize $n = 0$, $\mathbf{c}^{(0)} = [1, 0, 0, 0, 1, 0]$, and $\beta = 0.001$.
(2) For each $\mathbf{x}_j$ in the region of interest:
    (a) Compute $\widetilde{\mathbf{x}}_j = R\mathbf{x}_j - \mathbf{r}$ with entries of $R$ and $\mathbf{r}$ given by $\mathbf{c}^{(n)}$ as in (4.3) . Then compute $f(\widetilde{\mathbf{x}}_j)$.
    (b) Compute the error $e_j = f(\widetilde{\mathbf{x}}_j) - f_0(\mathbf{x}_j)$, the intensity gradient $(\partial f/\partial \tilde{x}, \partial f/\partial \tilde{y})$, and hence the partial derivatives $\partial e_j/\partial c_k$ for all $k$ as in (4.6).
    (c) Accumulate the results to the corresponding entries in $A$ and $\mathbf{b}$ as in (4.5).
(3) Compute $\triangle\mathbf{c} = (A + \beta I)^{-1}\mathbf{b}$ and update $\mathbf{c}^{(n+1)} = \mathbf{c}^{(n)} + \triangle\mathbf{c}$.
(4) If the error in (4.4) has decreased, go to Step (2); else increase $\beta$ by 100 times and compute a new $\triangle\mathbf{c}$.
(5) Continue iterating until either the error is below a threshold or a fixed number of iterations has been reached. Then return the $R$ and $\mathbf{r}$ corresponding to $\mathbf{c}^{(n+1)}$.
```

4.2. **Removing Motion Effects.** Since the framelet algorithm in §3 only handles translation between frames, in this subsection, we remove other motion effects in the frames so that each frame can be considered as a translation of the reference frame f_0, i.e. it can be viewed as the (i, j)th low resolution image with $0 \le i, j < 1$. Then we determine the index (s_k^x, s_k^y) for each $f_k \in \{f_k\}_{k \neq 0}$, and its corresponding displacement errors ϵ_k^x and ϵ_k^y.

Let $f \in \{f_k\}_{k \neq 0}$, by (4.2), $f_0(\mathbf{x}) \approx f(R\mathbf{x} - \mathbf{r}) = f[R(\mathbf{x} - R^{-1}\mathbf{r})]$. Thus $f(R(\cdot))$ can be viewed as a translation of f_0 with displacement vector $-R^{-1}\mathbf{r}$. Our task is to write

$$R^{-1}\mathbf{r} = \mathbf{u} + \frac{1}{2}\begin{bmatrix} s^x \\ s^y \end{bmatrix} + \frac{1}{2}\begin{bmatrix} \epsilon^x \\ \epsilon^y \end{bmatrix}, \tag{4.7}$$

where s^x, $s^y \in \{0, 1\}$ and both $|\epsilon^x|$ and $|\epsilon^y|$ are less than 1/2 (see (3.4)). Then $\hat{f}(\mathbf{x}) \equiv f(R(\mathbf{x} - \mathbf{u}))$ can be considered as the low-resolution image $\mathbf{g}_{s^x, s^y}$ with displacement errors (ϵ^x, ϵ^y). The following is the algorithm.

Algorithm 4. $(\hat{f}, s^x, s^y, \epsilon^x, \epsilon^y) \leftarrow$ *Register*(f, f_0): *Register frame f against the reference frame f_0.*

```
(1) Use Algorithm 3 to compute $(R, \mathbf{r}) \leftarrow$ Affine$(f, f_0)$.
(2) If the peak signal to noise ratio (PSNR) of $[f_0(\mathbf{x}) - f(R\mathbf{x} - \mathbf{r})]$ is less than $P_0$, then registration fails, return. Otherwise, compute $[\tilde{r}_1, \tilde{r}_2]^t = R^{-1}\mathbf{r}$.
(3) Let $\mathbf{u} \equiv [\ \lfloor \tilde{r}_1 + \frac{1}{4} \rfloor, \lfloor \tilde{r}_2 + \frac{1}{4} \rfloor\ ]^t$, then $[d_1, d_2] \equiv [\tilde{r}_1, \tilde{r}_2] - \mathbf{u}^t$ has entries in $[-\frac{1}{4}, \frac{3}{4})$.
(4) Let $[s^x, s^y] \equiv [\ \lfloor 2d_1 + \frac{1}{2} \rfloor, \lfloor 2d_2 + \frac{1}{2} \rfloor\ ]$, then $s_i^x, s_i^y \in \{0, 1\}$.
(5) Let $[\epsilon^x, \epsilon^y] \equiv [2d_1 - s^x, 2d_2 - s_i^y]$, then $|\epsilon_i^x|, |\epsilon_i^y| \le \frac{1}{2}$, and (4.7) holds.
(6) $\hat{f}(\mathbf{x}) \equiv f(R(\mathbf{x} - \mathbf{u}))$.
```

The threshold P_0 in Step (2) determines if $f(R\mathbf{x} - \mathbf{r})$ is close enough to $f_0(\mathbf{x})$ or else we discard the frame. We recall that PSNR for M-by-N color images with 8 bits for each red (r), green (g) and blue (b) channel is defined as:

$$\text{PSNR of } [\mathbf{u} - \mathbf{x}] = 10 \log_{10} \frac{255^2}{\frac{1}{3MN}\sum_{i-1}^{M}\sum_{j=1}^{N}\sum_{k=r,g,b}(u_{i,j}^{(k)} - x_{i,j}^{(k)})} \ (dB).$$

In the experiments, we set $P_0 = 25$dB.

4.3. **The Video Still Enhancing Algorithm.** Our video enhancing algorithm operates on one frame at a time. First we check if the incoming frame is useful or not by using Algorithm 4. If it is registered, we will incorporate it to enhance f_0 by using the high-resolution image reconstruction method in §3. More precisely, assume that the registered frame $\hat{f}$ is the (s^x, s^y)th low resolution image with displacement error (ϵ^x, ϵ^y), i.e.

$$\hat{f}(\cdot) = \widetilde{g}_{s^x,s^y}(\cdot,\cdot) = g_{s^x,s^y}(\cdot + \epsilon^x, \cdot + \epsilon^y),$$

cf (3.3). The method in §3 assumes that we have a complete set of low resolution images $\{\widetilde{g}_{i,j}\}_{i,j=0}^1$, whereas here we only have one. Our idea is to generate the missing low resolution images by downsampling the current high resolution approximation of f_0 with zero displacement errors, i.e. $\widetilde{\mathbf{g}}_{i,j} = D_{i,j}\mathbf{H}\mathbf{h}$ with $(\epsilon^x_{i,j}, \epsilon^y_{i,j}) = (0,0)$ for all $(i,j) \neq (s^x, s^y)$, see (2.12).

From (3.5), we know that,

$$\begin{aligned}
\mathbf{g} &= \widetilde{\mathbf{g}} - \mathbf{g}_\epsilon \\
&= \sum_{i,j=0}^{1} U_{i,j}\left[\widetilde{\mathbf{g}}_{i,j} - D_{i,j}\left(\sqrt{2}\epsilon^x_{i,j}\mathbf{T}_{0,1} + \sqrt{2}\epsilon^y_{i,j}\mathbf{T}_{1,0} + 2\epsilon^x_{i,j}\epsilon^y_{i,j}\mathbf{T}_{1,1}\right)\right]\mathbf{h} \\
&= \sum_{\substack{i,j=0,\\ (i,j)\neq(s^x,s^y)}}^{1} U_{i,j}D_{i,j}\mathbf{T}_{0,0}\mathbf{h} + U_{s^x,s^y}\left[\hat{\mathbf{f}} - D_{s^x,s^y}(\sqrt{2}\epsilon^x\mathbf{T}_{0,1} + \sqrt{2}\epsilon^y\mathbf{T}_{1,0} + 2\epsilon^x\epsilon^y\mathbf{T}_{1,1})\right]\mathbf{h}.
\end{aligned}$$

This leads to the following algorithm:

Algorithm 5. $\mathbf{h} \leftarrow$ *Update*$(\mathbf{h}, \hat{\mathbf{f}}, s^x, s^y, \epsilon^x, \epsilon^y)$: *Update the high resolution image* $\mathbf{h}$ *by a registered frame* $\hat{\mathbf{f}}$ *with parameters* $(s^x, s^y, \epsilon^x, \epsilon^y)$.

```
(1) Let $\hat{\mathbf{h}}^{(0)} = \mathbf{h}$ and $m = 0$
(2) Let $\mathbf{g}_\epsilon^{(m+1)} = U_{s^x,s^y}D_{s^x,s^y}(\sqrt{2}\epsilon^x\mathbf{T}_{0,1} + \sqrt{2}\epsilon^y\mathbf{T}_{1,0} + 2\epsilon^x\epsilon^y\mathbf{T}_{1,1})\hat{\mathbf{h}}^{(m)}$.
(3) If PSNR$[\hat{\mathbf{f}} - D_{s^x,s^y}\mathbf{g}_\epsilon^{(m+1)} - D_{s^x,s^y}\mathbf{T}_{0,0}\hat{\mathbf{h}}^{(m)}] < 50dB$, then $m+1 \to m$ and $\mathbf{h}_0^{(m)} = \hat{\mathbf{h}}^{(m-1)}$.
    Otherwise, $\hat{\mathbf{h}}^{(m)} \to \mathbf{h}$, end.
(4) Iterate on $\mathbf{h}_n^{(m)}$ until convergence:
    (a) $\mathbf{g}_{i,j} = \begin{cases} \hat{\mathbf{f}} - D_{s^x,s^y}\mathbf{g}_\epsilon^{(m)}, & (i,j) = (s^x,s^y), \\ D_{i,j}\mathbf{T}_{0,0}\mathbf{h}_n^{(m)}, & \text{else.} \end{cases}$
    (b) $\mathbf{g} = \sum_{i,j=0}^{1} U_{i,j}\mathbf{g}_{i,j}$.
    (c) $\mathbf{h}_{n+1}^{(m)} = \mathbf{T}_{0,0}^*\mathbf{g} + \sum_{\substack{i,j=0\\(i,j)\neq(0,0)}}^{2} \mathbf{T}_{i,j}^*\mathcal{A}_J^*\mathcal{T}\mathcal{A}_J\mathbf{T}_{i,j}\mathbf{h}_n^{(m)}$.
(5) Set $\hat{\mathbf{h}}^{(m)} = \mathbf{h}_n^{(m)}$ when converge, and go back to Step (2).
```

Remark 4.1.

(1) We stop Step (4) when the PSNR of the difference of two consecutive $\mathbf{g}$ is greater than 40dB.
(2) In our experiments, we set $J = 3$. Step (3) usually converges in 2–3 iterations, while Step (4) in 3–6 iterations when $m = 1$ and the number of iterations decreases with increasing m.

In the following we give the complete algorithms of our method. Given a reference frame f_0, it is conceivable that the frames taken just before or after f_0 will give the most relevant information regarding f_0. Thus we write our algorithm for a sequence of $2K$ frames $\{f_k\}_{k=-K}^{K}$ that are taken just before and after the reference frame f_0.

Algorithm 6. *Resolution Enhancement for Video Clips I.*

```
(1) Obtain an initial guess of the high resolution image $\mathbf{h}$ by using bilinear interpolation on $f_0$.
(2) for $j = -1, 1, -2, 2, \cdots, -K, K$:
    (a) Apply Algorithm 4 to get $(\hat{f}_j, s_j^x, s_j^y, \epsilon_j^x, \epsilon_j^y) \leftarrow$ Register$(f_j, f_0)$.
    (b) If registration is successful, use Algorithm 5 to update $\mathbf{h} \leftarrow$ Update$(\mathbf{h}, \hat{\mathbf{f}}_j, s_j^x, s_j^y, \epsilon_j^x, \epsilon_j^y)$.
```

Algorithm 6 uses the new information from good frames to update f_0. Its advantage is that it chooses the good candidate frames automatically and there is no need to determine the number of frames to be used in advance. It applies each good frame once. The next algorithm applies each good frame twice, one forward and one backward. The results are better but it is slower.

Algorithm 7. *Resolution Enhancement for Video Clips II.*

```
(1) Obtain an initial guess of $\mathbf{h}$ by using bilinear interpolation on $f_0$. Set $\ell = 0$.
(2) for $j = -1, 1, -2, 2, \cdots, -K, K$:
    (a) Apply Algorithm 4 to get $(\hat{f}_j, s_j^x, s_j^y, \epsilon_j^x, \epsilon_j^y) \leftarrow$ Register$(f_j, f_0)$.
    (b) If registration is successful, then, $\mathbf{p}_l \leftarrow (\hat{\mathbf{f}}_j, s_j^x, s_j^y, \epsilon_j^x, \epsilon_j^y)$, apply Algorithm 5 to update
        $\mathbf{h} \leftarrow$ Update$(\mathbf{h}, \mathbf{p}_l)$ and $\ell \leftarrow \ell + 1$.
(3) for $i = \ell - 1, \cdots, 0$:
        Apply Algorithm 5 to update $\mathbf{h} \leftarrow$ Update$(\mathbf{h}, \mathbf{p}_i)$.
```

One can easily extend our algorithms to color images. In color imaging, it is well-known that the intensity component plays the most important role amongst all color components. Thus given a color image, we first change it from the RGB color space to the YCrCb color space, see [8]. Then we apply our algorithms to each of the components in the YCrCb space simultaneously. More precisely, we have $\mathbf{f} = (\mathbf{f}^Y, \mathbf{f}^{Cr}, \mathbf{f}^{Cb})$ in the algorithms. However we use the Y (the intensity) component for the stopping criteria, e.g. Step (3) of Algorithm 5 will stop if

$$\text{PSNR}\{\hat{\mathbf{f}}^Y - D_{s^x,s^y}[\mathbf{g}_\epsilon^{(m+1)}]^Y - D_{s^x,s^y}\mathbf{T}_{0,0}[\hat{\mathbf{h}}^{(m)}]^Y\} > 50\text{dB}.$$

5. Experimental Results

In this section, we implement and test our resolution enhancement algorithms for two video clips. The first one is filmed by us by panning our video camcorder over some books on a table. The clip is in CIF format with size 352-by-288 and can be downloaded at [17]. In the five second video clip, we choose the 100th frame as our reference frame f_0, see Figure 2. Figure 3 gives the first guess of the high resolution image of f_0, which is obtained by the bilinear interpolation on f_0. It is of size 704-by-576. We let $K = 10$, i.e. we use the 91th to 110th frames to improve the resolution of f_0. The results of Algorithms 6 and 7 are shown in Figures 4 and 5 respectively.

The alignment parameters of frames $\{f_k\}_{k=90}^{110}$ using Algorithm 4 are listed in Table 1. The first column is the index of the frame; the second and the third columns list low resolution image index (s^x, s^y) and displacement error (ϵ^x, ϵ^y) for each frame; and the fourth column indicates whether the frame is close to the reference frame f_{100}. Note that frames f_{94}, f_{95}, and f_{109} are discarded.

From the resulting high resolution images, the words in the title of the books such as "Linear Analysis and Differential Equations", "Programmer's Guide", and "Classical Fourier Transforms" are clearly discernible. This is very difficult to do from the original frame or from the video clip themselves. Moreover the number "98" on the yellow book "Box Spline" is much clearer in Figure

Table 1. Alignment Results from Algorithm 4 for Book Clip

Frame Index	(s^x, s^y)	(ϵ^x, ϵ^y)	$f_0(\mathbf{x}) \approx f(R\mathbf{x}+\mathbf{r})$
101	(1,0)	(-0.237, 0.150)	Yes
99	(0,0)	(-0.409,-0.180)	Yes
102	(0,0)	(-0.153,-0.162)	Yes
98	(0,1)	(0.464, 0.314)	Yes
103	(0,0)	(0.437, 0.216)	Yes
97	(0,1)	(0.178, 0.402)	Yes
104	(1,0)	(0.478,-0.101)	Yes
96	(1,0)	(0.453,-0.343)	Yes
105	(1,0)	(-0.465, 0.014)	Yes
95	...	...	**No**
106	(0,0)	(-0.271,-0.129)	Yes
94	...	...	**No**
107	(1,0)	(-0.021,-0.036)	Yes
93	(1,0)	(-0.335,-0.449)	Yes
108	(0,1)	(0.044, 0.420)	Yes
92	(0,0)	(-0.057,-0.336)	Yes
109	...	...	**No**
91	(0,0)	(-0.049,-0.456)	Yes
110	(1,0)	(-0.453, 0.052)	Yes
90	(0,1)	(-0.136, 0.093)	Yes

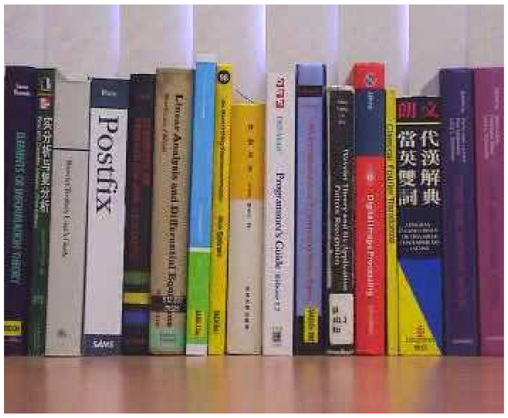

Figure 2. The 100th Low Resolution Frame

5 than in Figure 4. To compare them more clearly, in Figure 6 we give the zoom-in results for this part of the image. Note that it is impossible to read this number from the video clip or just by using interpolation on the reference frame.

We note that in [5], we have used a rudimentary version of the algorithm proposed here to enhance the same video clip. Although the numerical results are similar, one major difference between the two algorithms is that in [5], symmetirc boundary condition is used. The resulting blurring matrix $\mathbf{T}_0$ may then be singular. Hence, one cannot establish convergence theorem like that in Theorem 2.2 here.

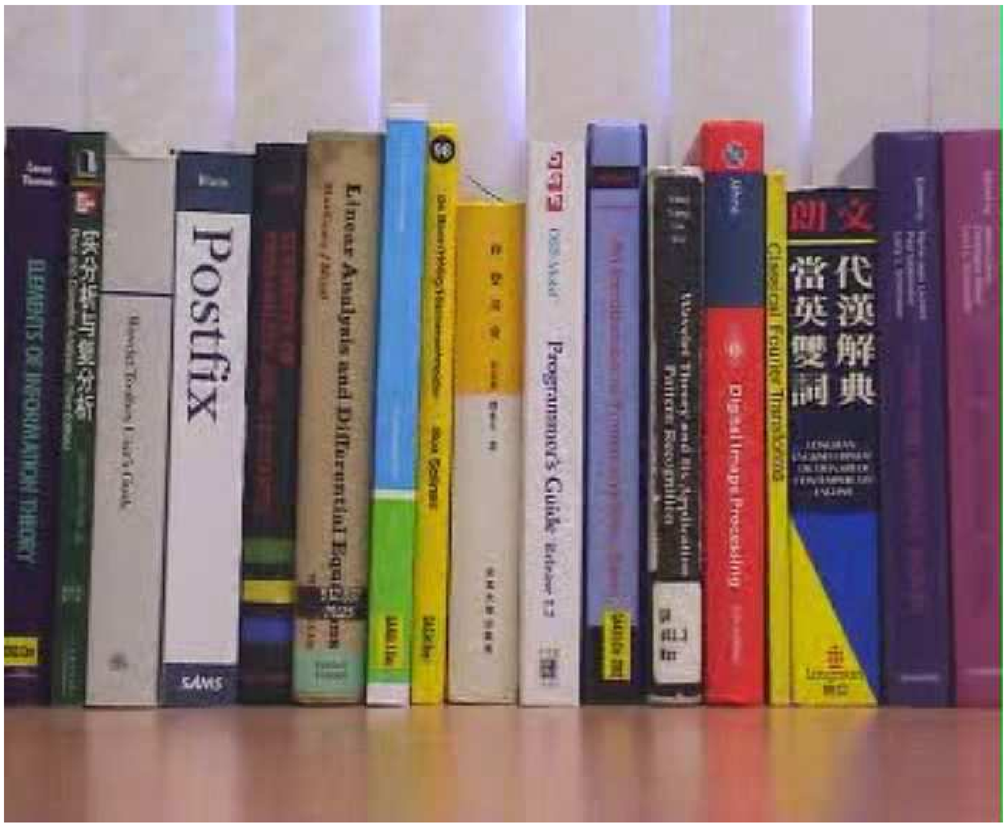

Figure 3. First Guess of the High Resolution Image

Next we test our algorithm on another video clips given in [10], with the courtesy of University of Maryland at College Park. The video clip is a box moved by a person. The frame rate is 15 and the resolution is 512-by-384. We cropped a central part of the images with resolution 96-by-128 manually for the demonstration of our algorithm.

In the 1.5 seconds of the video clip, we choose the 11th frame as our reference frame f_0, see Figure 7. Figure 8 gives the first guess of the high-resolution image of f_0 by the bilinear interpolation. It is of size 192-by-256. We let $K = 10$, i.e. we use the 1st to the 19th frames to improve the resolution of f_0. The results are shown in Figure 8. Note that the third row of the texts is much clearer using our algorithm than by the interpolation. The alignment parameters for this clip are listed in Table 2, which shows that all frames are used for this video clip.

From the two experiments, we see that the image contents of the reference frames have been improved much by our algorithms as it can reveal information that are not discernible in the video clips or by simple interpolation.

6. Conclusions

In this paper, we propose algorithms to generate high resolution images from video clips. The results show that the reconstructed images can disclose information that cannot be found either in the original frames or in the video clips themselves. Although we have restricted our discussions to enhancement ratio L equal to 2 (see the definition of L in §2.1), it is straightforward to modify

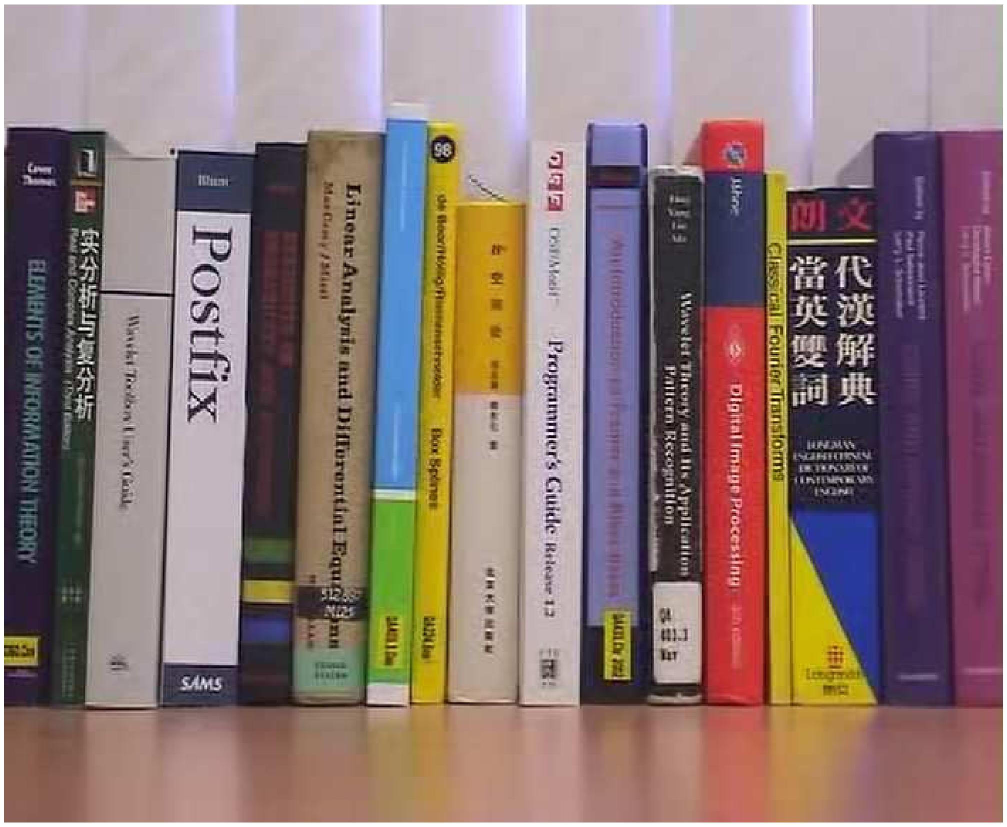

FIGURE 4. Reconstructed High Resolution Image using Algorithm 6

TABLE 2. Alignment Results of Algorithm 4 for Box Clip

Frame Index	(s^x, s^y)	(ϵ^x, ϵ^y)	$f_0(\mathbf{x}) \approx f(R\mathbf{x}+\mathbf{r})$
12	(1,0)	(-0.279,-0.475)	Yes
10	(1,1)	(0.401,-0.491)	Yes
13	(0,1)	(-0.040,-0.071)	Yes
9	(1,1)	(-0.238, 0.406)	Yes
14	(1,0)	(0.404, 0.337)	Yes
8	(0,0)	(0.029, 0.022)	Yes
15	(1,0)	(-0.342,-0.084)	Yes
7	(1,0)	(0.335, 0.307)	Yes
16	(1,1)	(-0.236, 0.438)	Yes
6	(1,1)	(-0.333,-0.386)	Yes
17	(1,1)	(0.331,-0.492)	Yes
5	(0,1)	(0.161,-0.158)	Yes
18	(0,0)	(-0.278,-0.296)	Yes
4	(1,0)	(-0.313,-0.460)	Yes
19	(0,1)	(0.299, 0.098)	Yes
3	(1,0)	(0.062,-0.004)	Yes
2	(1,1)	(0.177,-0.466)	Yes
1	(0,1)	(0.173,-0.080)	Yes

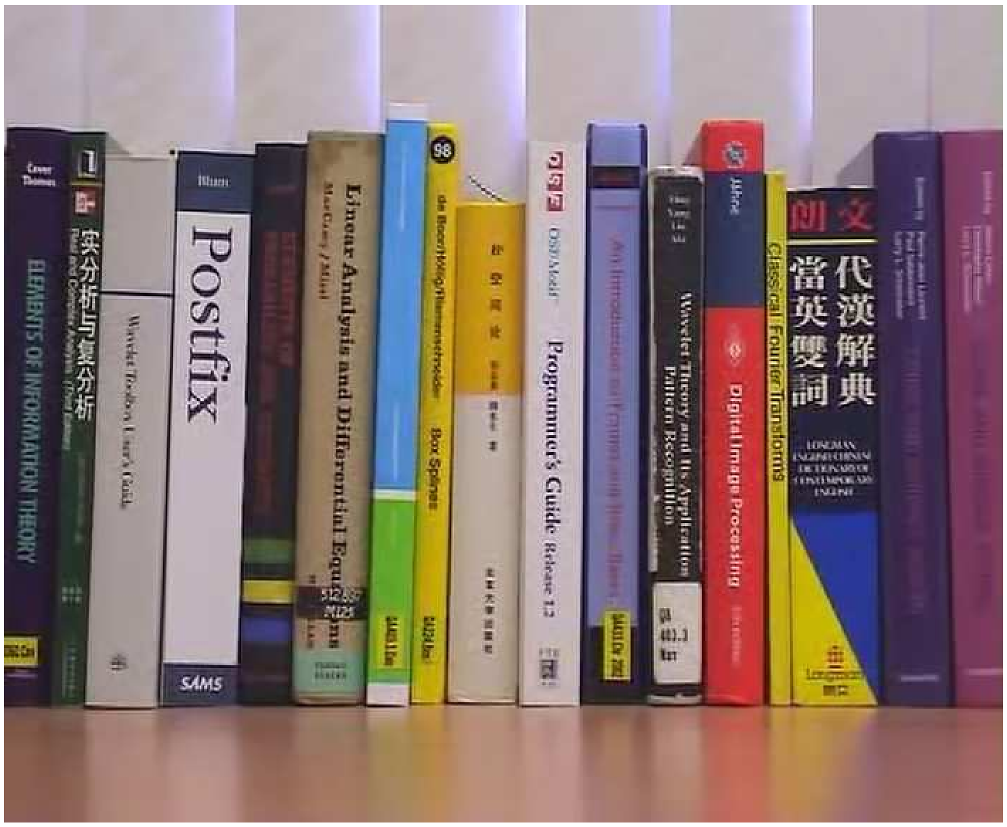

FIGURE 5. Reconstructed High Resolution Image using Algorithm 7

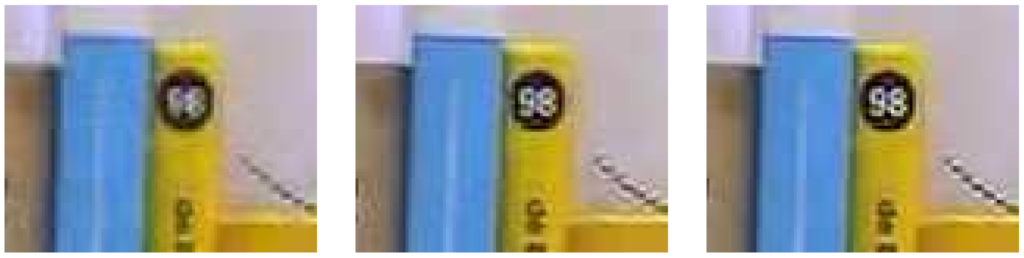

FIGURE 6. Zoom-in of Figure 3 (First Guess), Figure 4 (Algorithm 6) and Figure 5 (Algorithm 7)

our algorithm to larger L. For example, (3.4) will have to be changed to:

$$0 \leq |\epsilon_{i,j}^x|, |\epsilon_{i,j}^y| < \frac{1}{L}, \quad i,j = 0, 1, \cdots, L-1.$$

Unlike the works in [11, 16], where motion vectors are exploited for the resolution enhancement, only affine model for translation is considered here. Therefore, frames that are far away from the reference frame may not be registered by our method, even though they may share a big portion in common with the reference frame. Our future work will focus on more complicated scenarios to include motion models other than affine translations.

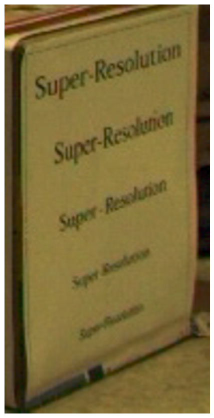

Figure 7. The 11th Low Resolution Frame

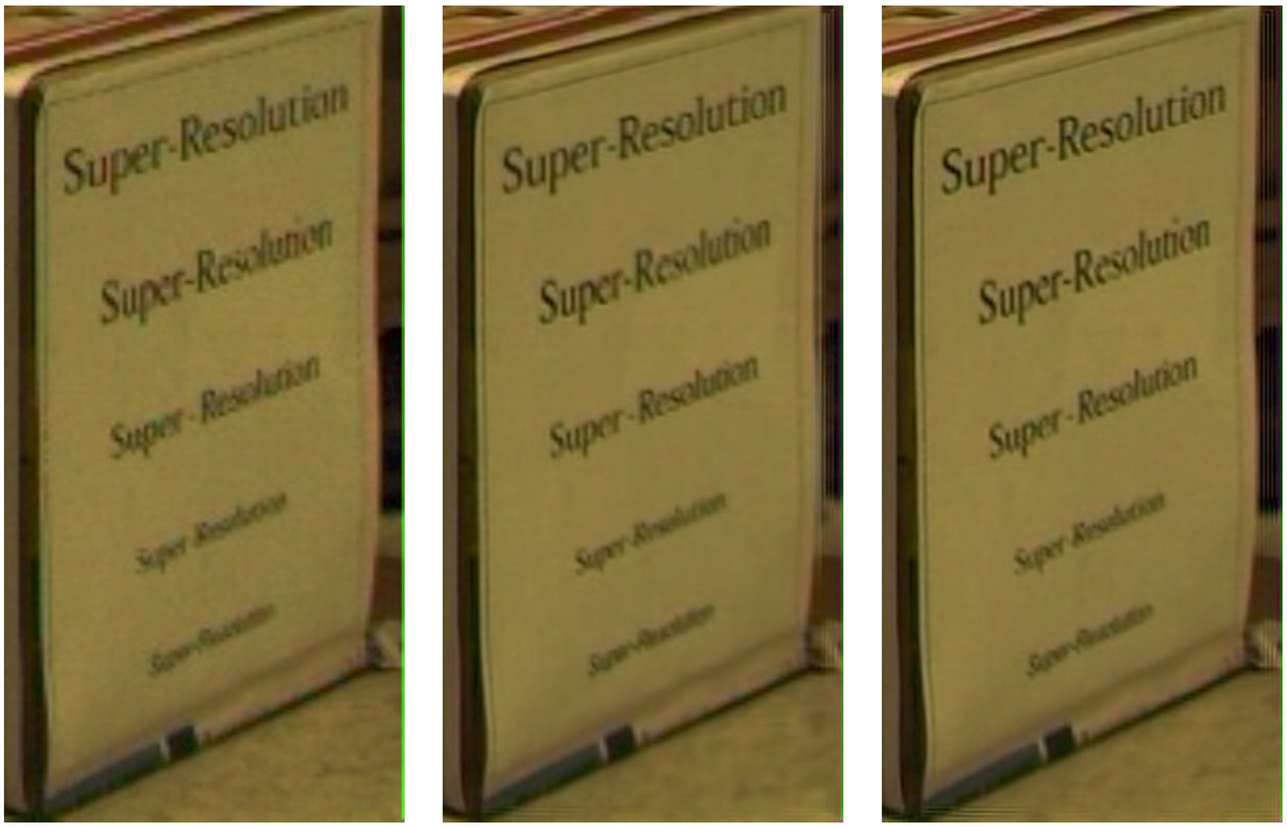

Figure 8. First Guess by Bi-linear Interpolation (Left), High Resolution Image from Algorithm 6 (Center) and from Algorithm 7 (Right)

References

[1] N. Bose and K. Boo. High-resolution image reconstruction with multisensors, *International Journal of Imaging Systems and Technology*, 9:294–304, 1998.

[2] A. Chai and Z. Shen, Deconvolution: A wavelet frame approach, *preprint*, 2005.

[3] R. Chan, T. Chan, L. Shen and Z. Shen, Wavelet algorithms for high-resolution image reconstruction, *SIAM Journal on Scientific Computing*, 24(4):1408–1432, 2003.

[4] R. Chan, S.D. Riemenschneider, L. Shen and Z. Shen, Tight frame: an efficient way for high-resolution image reconstruction, *Applied and Computational Harmonic Analysis*, 17:91–115, 2004.

[5] R. Chan, Z. Shen and T. Xia, Resolution enhancement for video clips : tight frame approach, *Proceedings of IEEE International Conference on Advanced Video and Signal Based Surveillance 2005*, pp. 406-410, Sept. 2005, Como, Italy.

[6] I. Daubechies, M. Defrise and C. De Mol, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, *Comm. Pure Appl. Math.*, 57:1413–1457, 2004.

[7] D. Donoho and I. Johnstone, Ideal spatial adaptation by wavelet shrinkage, *Biometrika*, 81:425–455, 1994.

[8] K. Jack, *Video demystified : a handbook for the digital engineer*, HighText Publications, San Diego, Calif., 1996.

[9] B. Jähne, *Digital image processing*, 5th Edition, Springer-Verlag, Berlin Heidelberg, 2002.

[10] H. Ji and C. Fermuller, *Super-resolution reconstruction from extended video sequences*, preprint, submitted to EVVC 2006.

[11] S. Lertrattanapanich and N. K. Bose, Latest results on high-resolution reconstruction from video sequences, *Technical Report of IEICE, DSP99-140, The Institution of Electronic, Information and Communication Engineers*, Japan, pp. 59–65, December 1999.

[12] M. Ng, R. Chan, and W. Tang, A fast algorithm for deblurring models with Neumann boundary conditions, *SIAM Journal on Scientific Computing*, 21:851–866, 2000.

[13] W.H. Press et al., *Numerical recipes in C++ : The art of scientific computing*, 2nd Edition, Cambridge Univ. Press, Cambridge, England, 1992.

[14] A. Ron and Z. Shen, Affine systems in $\mathcal{L}^2(\mathbb{R}^d)$: the analysis of the analysis operator, *Journal of Functional Analysis*, 148:408–447, 1997.

[15] R. Szeliski, Video mosaics for virtual environments, *IEEE Computer Graphics and Applications*, 22–30, March 1996.

[16] B.C. Tom and A. K. Katsaggelos, Resolution enhancement of monochrome and color video using motion compensation, *IEEE Transactions on Image Processing*, 10(2):278–287, 2001.

[17] Video clip and full results downloadable at `http://www.math.cuhk.edu.hk/~rchan/paper/csx`.