Chapter 1

A FRAMELET-BASED ALGORITHM FOR VIDEO ENHANCEMENT

Raymond H. Chan

Department of Mathematics, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong.

rchan@math.cuhk.edu.hk

Yiqiu Dong

START-Project, Institute of Mathematics and Scientific Computing, University of Graz, Heinrichstrasse 36, A-8010 Graz, Austria.

yiqiu.dong@uni-graz.at

Zexi Wang

Department of Finance and Management Science, Norwegian School of Economics and Business Administration, Helleveien 30, NO-5045 Bergen, Norway.

zexi.wang@nhh.no

Abstract Video clips are made up of many still frames. Most of the times, the frames are small perturbations of their neighboring frames. Recently, we proposed a framelet-based algorithm to enhance the resolution of any frames in a video clip by solving it as a super-resolution image reconstruction problem. In this paper, we extend the algorithm to video enhancement, where we compose a high-resolution video from a low-resolution one. An experimental result of our algorithm on a real video clip is given to illustrate the performance.

Keywords: high-resolution image reconstruction, video enhancement.

Introduction

High-resolution images are useful in remote sensing, surveillance, military imaging, and medical imaging, see for instances Mather04; LKC00;

CZ03. However, high-resolution images are more expensive to obtain compared to low-resolution ones which can be obtained from an array of inexpensive low-resolution sensors. Therefore, there has been much interest in reconstructing high-resolution images from low-resolution ones that are small perturbation of each other, see KS93; NKS95; SS96; EST97; YP03; MMK03. One approach is based on the maximum likelihood technique using the expectation maximization algorithm to seek the high resolution image CHA96. Another approach is the regularization method in HBBAW98 which was based on the total squared error between the observed low-resolution images and the predicted low-resolution images. The predicted images are the results of projecting the high-resolution image estimate through the observation model. The framelet algorithm proposed in CRSS1; CRSS2 is different from these methods. It applies the unitary extension principle in RS97 to form a system of tight frame filters. In this approach, there is only matrix-vector multiplication, and no need for solving a minimization problem. Recently, it was shown that this framelet algorithm, which is an iterative algorithm, indeed converges to a minimizer of a variational problem CCS08.

Video clips are made up of many still frames (about 25 to 30 frames per second), and the scene usually does not change much from one frame to the next. Thus given a reference frame, its nearby frames can be considered as its small perturbations, and we can make use of them to get a high-resolution image of the reference frame. More precisely, consider a sequence of frames $\{f_k\}_{k=-K}^{K}$ in a video clip, where k increases with the time when the frame f_k is taken. Let f_0 be the reference frame we want to enhance its resolution. The frames $\{f_k\}_{k\neq 0}$ can be considered as small spatial perturbations of f_0. Then, we can use the framelet algorithm proposed in CRSS1 to improve the resolution of f_0. Such a still enhancement algorithm is given in CSX07.

The goal of this paper is to extend the algorithm in CSX07 to video enhancement, where high-resolution video streams are constructed from low-resolution ones. The paper is organized as follows. In Section 1, we introduce the framelet algorithm for the high-resolution image reconstruction given in CRSS1. In Section 2, we describe the algorithm proposed in CSX07 for enhancing video stills. Then in Section 3, we extend it to video enhancement and apply the resulting algorithm on a real video to enhance the video resolution. Conclusion is given in Section 4.

In this paper, we use bold-face characters to indicate vectors. If f represents an image $f(x,y)$, $\mathbf{f}$ represents the column vector constructed by raster scanning of f row by row.

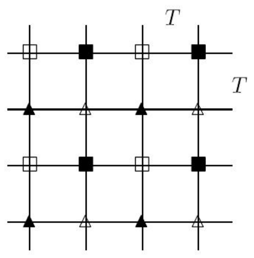


Figure 1.1. The model of obtaining g by interlacing pixels from $g_{i,j}$. (□ : $g_{0,0}$ pixels; ■ : $g_{0,1}$ pixels; ▲ : $g_{1,0}$ pixels; △: $g_{1,1}$ pixels.)

1. High-Resolution Image Reconstruction

The Model

Here we briefly recall the high-resolution image reconstruction model introduced in BB98. For more details, please refer to the paper. Let h be a piecewise continuous function measuring the intensity of a scene. An image of h at sampling resolution T can be modeled by the integral

$$h(n_1,n_2) \equiv \frac{1}{T^2}\int_{(n_2-\frac{1}{2})T}^{(n_2+\frac{1}{2})T}\int_{(n_1-\frac{1}{2})T}^{(n_1+\frac{1}{2})T} h(x,y)dxdy, \quad n_1,n_2\in\mathbb{Z}. \tag{1.1}$$

Here (n_1,n_2) are the pixel locations. High-resolution image reconstruction refers to the construction of an image with sampling resolution T by using K^2 low-resolution images of sampling resolution KT, where K is a positive integer. In this paper, we only consider $K=2$. Larger value of K can be considered similarly, but with more complicated notations.

When $K=2$, we are given four low-resolution images, $g_{0,0}, g_{0,1}, g_{1,0}, g_{1,1}$ of sampling resolution $2T$, sampling at

$$g_{i,j}(n_1',n_2') = \frac{1}{4T^2}\int_{(2(n_2'-\frac{1}{2})+j)T}^{(2(n_2'+\frac{1}{2})+j)T}\int_{(2(n_1'-\frac{1}{2})+i)T}^{(2(n_1'+\frac{1}{2})+i)T} h(x,y)dxdy, \tag{1.2}$$

where $i,j=0,1$. The locations $(0,0)$, $(0,1)$, $(1,0)$ and $(1,1)$ are the sensor positions.

A straightforward way to form an image g of sampling resolution T is to interlace the four low-resolution images, i.e.

$$g(n_1,n_2) = g_{i,j}(n_1',n_2'), \tag{1.3}$$

where $i = n_1 \bmod 2$, $j = n_2 \bmod 2$, $n_1' = \lfloor n_1/2\rfloor, n_2' = \lfloor n_2/2\rfloor$, see Figure 1.1. The function g is called the *observed high-resolution image.*

Note that g is not equal to the desired image h in (1.1) but is a good approximation of it. If we assume $h(x,y)$ is constant in the sampling region

$$[(n_1-.5)T,(n_1+.5)T)\times[(n_2-.5)T,(n_2+.5)T),$$

for all $n_1,n_2\in\mathbb{Z}$, (i.e. $h(x,y)\equiv h(n_1,n_2)$ there), then by (1.1)–(1.3), we can easily prove that

$$\begin{aligned} g(n_1,n_2)=&\frac{1}{4}[\frac{1}{4}h(n_1-1,n_2-1)+\frac{1}{2}h(n_1-1,n_2)+\frac{1}{4}h(n_1-1,n_2+1)\\ &+\frac{1}{2}h(n_1,n_2-1)+h(n_1,n_2)+\frac{1}{2}h(n_1,n_2+1)\\ &+\frac{1}{4}h(n_1+1,n_2-1)+\frac{1}{2}h(n_1+1,n_2)+\frac{1}{4}h(n_1+1,n_2+1)].\end{aligned}$$

In matrix form, it is

$$\mathbf{g}=H_{0,0}\mathbf{h}, \tag{1.4}$$

where $H_{0,0}=H_0\otimes H_0$ with H_0 being the matrix representation of the discrete convolution (i.e. Toeplitz form) with kernel $h_0=[1/4,1/2,1/4]$.

To obtain a better high-resolution image than $\mathbf{g}$, one will have to solve $\mathbf{h}$ from (1.4). It is an ill-posed problem where many methods are available. One approach is the framelet method in CRSS1 that we are going to describe next.

Framelet-Based HR Image Reconstruction

Here we briefly recall the algorithm in CRSS1 and refer the reader to the paper for more details. The convolution kernel h_0 is a low-pass filter. By applying the unitary extension principle in RS97, h_0 together with the following high-pass filters form a tight framelet system:

$$h_1=[\frac{\sqrt{2}}{4},0,-\frac{\sqrt{2}}{4}],\quad h_2=[-\frac{1}{4},\frac{1}{2},-\frac{1}{4}]. \tag{1.5}$$

Define

$$H_{i,j}=H_i\otimes H_j,\quad 0\le i,j\le 2,$$

where H_i is the discrete convolution matrix with kernel h_i. The perfect reconstruction formula for the framelet systems gives

$$\mathbf{h}=\sum_{i,j=0}^{2}H_{i,j}^{*}H_{i,j}\mathbf{h},$$

see CRSS1. Based on (1.4) and substituting $H_{0,0}\mathbf{h}$ by $\mathbf{g}$, we have

$$\mathbf{h}=H_{0,0}^{*}\mathbf{g}+\sum_{\substack{i,j=0\\(i,j)\neq(0,0)}}^{2}H_{i,j}^{*}H_{i,j}\mathbf{h}. \tag{1.6}$$

$$\begin{tikzcd}
 & H_{0,0}\mathbf{w} \arrow[r, "\mathcal{T}_{v-1}"] & {} \arrow[r, "H^*_{0,0}"] & H^*_{0,0}\mathcal{T}_{v-1}(H_{0,0}\mathbf{w}) \arrow[dr] & \\
\mathbf{w} \arrow[ur, "H_{0,0}"] \arrow[r, "H_{0,1}"] \arrow[ddr, "H_{2,2}"'] & H_{0,1}\mathbf{w} \arrow[r, "\mathcal{D}_\lambda"] & {} \arrow[r, "H^*_{0,1}"] & H^*_{0,1}\mathcal{D}_\lambda(H_{0,1}\mathbf{w}) \arrow[r] & \mathcal{T}_v\mathbf{w} \\
 & \vdots & \vdots & \vdots & \\
 & H_{2,2}\mathbf{w} \arrow[r, "\mathcal{D}_\lambda"] & {} \arrow[r, "H^*_{2,2}"] & H^*_{2,2}\mathcal{D}_\lambda(H_{2,2}\mathbf{w}) \arrow[uur] &
\end{tikzcd}$$

Figure 1.2. The operator $\mathcal{T}_v$ defined recursively with $\mathcal{T}_0 = I$, the identity.

Images are usually contaminated with noise, which are of high-frequency in nature. Since except for $H_{0,0}$ the others filter matrices are all high-pass, the noise is magnified in the second term of (1.6). We can use a threshold operator $\mathcal{D}_\lambda$ to remove the noise. The iteration, in matrix terms, thus becomes

$$\mathbf{h}^{(n+1)} = H^*_{0,0}\mathbf{g} + \sum_{\substack{i,\, j\, =\, 0 \\ (i,\, j)\, \neq\, (0,\, 0)}}^{2} H^*_{i,j}\mathcal{D}_\lambda(H_{i,j}\mathbf{h}^{(n)}), \quad n = 0, 1, \ldots,$$

where $\mathbf{h}^{(0)}$ is the initial guess. Here we use Donoho's soft thresholding operator Donoho95:

$$\mathcal{D}_\lambda(\mathbf{x}) = (t_\lambda(x_1), \cdots, t_\lambda(x_L))^\top,$$

where $t_\lambda(x) = \operatorname{sgn}(x)\max(|x| - \lambda, 0)$, $\lambda = 2\sigma\sqrt{\log L}$, L is the length of the vector $\mathbf{x}$, and σ is the variance of the noise estimated numerically by the method in Donoho95.

However, to avoid too many high-frequency components being removed, we use wavelet packets to further decompose the high-frequency components before doing the thresholding. In essence, we replace the operator $\mathcal{D}_\lambda$ by the recursively-defined $\mathcal{T}_v$ shown in Figure 1.2. The operator $\mathcal{T}_v$ will first decompose $\mathbf{w}$ until the level v and then threshold all the coefficients except the low-frequency ones on level v. In matrix terms, we have

$$\mathbf{h}^{(n+1)} = H^*_{0,0}\mathbf{g} + \sum_{\substack{i,\, j\, =\, 0 \\ (i,\, j)\, \neq\, (0,\, 0)}}^{2} H^*_{i,j}\mathcal{T}_v(H_{i,j}\mathbf{h}^{(n)}), \quad n = 0, 1, \ldots. \tag{1.7}$$

HR Reconstruction with Displacement Errors

Before we can apply the algorithm in Section 1.0 to improve video quality, there is one problem we have to overcome. In enhancing videos, the frames in the video may not be aligned exactly by length T as in (1.3) or in Figure 1.1. For example, relative to the reference frame, a nearby frame may have moved in the x-direction by a distance of ℓT where $\ell = n + r$, with $n \in \mathbb{Z}$ and $|r| < 1$. In that case, if we want to apply the algorithm in the last section, we can first shift the frame back by $nT = (n/2)(2T)$ and then consider the shifted frame as a displaced frame of the reference frame with displacement error equals $r/2$. The displacement error can then be corrected by framelet systems as follows. We refer the readers to CRSS1 for more details.

Define the 2D downsampling and upsampling matrices $D_{i,j} = D_j \otimes D_i$ and $U_{i,j} = U_j \otimes U_i$, where $D_i = I_M \otimes \mathbf{e}_i^\top$ and $U_i = I_M \otimes \mathbf{e}_i$ $(i = 0, 1)$, I_M is the identity of size M, $\mathbf{e}_0 = (1, 0)^\top$ and $\mathbf{e}_1 = (0, 1)^\top$. Here M-by-M is the resolution of the low-resolution frame. Then we have

$$\mathbf{g}_{i,j} = D_{i,j}\mathbf{g} \quad \text{and} \quad \mathbf{g} = \sum_{i,j=0}^{1} U_{i,j}\mathbf{g}_{i,j}. \tag{1.8}$$

As mentioned above, in practice, what we obtained is a shifted version of $\mathbf{g}_{i,j}$, i.e. we have $\tilde{\mathbf{g}}_{i,j}(\cdot,\cdot) \equiv \mathbf{g}_{i,j}(\cdot + \epsilon^x_{i,j}, \cdot + \epsilon^y_{i,j})$, where $0 \le |\epsilon^x_{i,j}| < 0.5$, $0 \le |\epsilon^y_{i,j}| < 0.5$, $0 \le i, j \le 1$. The parameters $\epsilon^x_{i,j}$ and $\epsilon^y_{i,j}$ are called *the displacement errors*. As in (1.4) and (1.8), the observed low-resolution image $\tilde{\mathbf{g}}_{i,j}$ can be considered as the down-sample of $\mathbf{h}$ after it has passed through a filter corresponding to the 2D framelet filter matrix $H = H(\epsilon^y_{i,j}) \otimes H(\epsilon^x_{i,j})$, where $H(\epsilon)$ denotes the 1D filter $\frac{1}{2}[\frac{1}{2} - \epsilon, 1, \frac{1}{2} + \epsilon]$. More precisely, we have

$$\tilde{\mathbf{g}}_{i,j} = D_{i,j}H\mathbf{h}.$$

Then $\mathbf{g}_{i,j}$ can be obtained from $\tilde{\mathbf{g}}_{i,j}$ via

$$\mathbf{g}_{i,j} = D_{i,j}H_{0,0}\mathbf{h} = D_{i,j}[H - (H - H_{0,0})]\mathbf{h} = \tilde{\mathbf{g}}_{i,j} - D_{i,j}(H - H_{0,0})\mathbf{h}.$$

Hence we have

$$\mathbf{g} = \sum_{i,j=0}^{1} U_{i,j}\mathbf{g}_{i,j} = \sum_{i,j=0}^{1} U_{i,j}[\tilde{\mathbf{g}}_{i,j} - D_{i,j}(H - H_{0,0})\mathbf{h}] = \tilde{\mathbf{g}} - (H - H_{0,0})\mathbf{h}. \tag{1.9}$$

Substituting (1.9) into (1.7), we arrive at the 2D image resolution enhancement formula:

$$\mathbf{h}^{(n+1)} = H^*_{0,0}[\tilde{\mathbf{g}} - (H - H_{0,0})\mathbf{h}^{(n)}] + \sum_{\substack{i,\, j\, =\, 0 \\ (i,\, j) \neq (0,\, 0)}}^{2} H^*_{i,j}\mathcal{T}_v(H_{i,j}\mathbf{h}^{(n)}). \tag{1.10}$$

We depict this algorithm graphically in Figure 1.3.

$$\begin{tikzcd}
 & H_{0,0}\mathbf{h}^{(n)} \arrow[rr, dashed, "H_{0,0}^*"] & & H_{0,0}^*[\tilde{\mathbf{g}} - (H - H_{0,0})\mathbf{h}^{(n)}] \arrow[dr] & \\
\mathbf{h}^{(n)} \arrow[ur, "H_{0,0}"] \arrow[r, "H_{0,1}"] \arrow[ddr, "H_{2,2}"'] & H_{0,1}\mathbf{h}^{(n)} \arrow[r, "\mathcal{T}_v"] & {} \arrow[r, "H_{0,1}^*"] & H_{0,1}^*\mathcal{T}_v(H_{0,1}\mathbf{h}^{(n)}) \arrow[r] & \mathbf{h}^{(n+1)} \\
 & \vdots & \vdots \quad \vdots & \vdots & \\
 & H_{2,2}\mathbf{h}^{(n)} \arrow[r, "\mathcal{T}_v"] & {} \arrow[r, "H_{2,2}^*"] & H_{2,2}^*\mathcal{T}_v(H_{2,2}\mathbf{h}^{(n)}) \arrow[uur] &
\end{tikzcd}$$

Figure 1.3. Framelet-based resolution enhancement algorithm for 2D images, see (1.10).

2. 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 higher resolution images of any frame in the video by exploiting the high redundancy between the nearby frames. 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. We aim to improve the resolution of the reference frame f_0 by incorporating information from frames $\{f_k\}_{k\neq 0}$. Without loss of generality, we can assume that f_0 is the low-resolution image at the (0,0) sensor position without any displacement error.

An algorithm for video still enhancement is given in CSX07 which is an adaptation of the algorithm in (1.10). Basically, we have to tackle the following issues:

1 for each frame f_k, we have to estimate its sensor position and displacement errors with respect to f_0, and

2 it may be that not all low-resolution images at all sensor positions are available in the video.

Here we recall the ways we handled these issues in CSX07.

Estimating the Motion Parameters

For computational efficiency, we assume that the frames $\{f_k\}_{k\neq 0}$ are related to f_0 by an affine transform, i.e.

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

where $\mathbf{x}$ are the coordinates of the pixels in the region of interest. Denote

$$R_k\mathbf{x} - \mathbf{r}_k \equiv \begin{bmatrix} c_0^{(k)} & c_1^{(k)} \\ c_3^{(k)} & c_4^{(k)} \end{bmatrix}\mathbf{x} + \begin{bmatrix} c_2^{(k)} \\ c_5^{(k)} \end{bmatrix} = \begin{bmatrix} c_0^{(k)} & c_1^{(k)} & c_2^{(k)} \\ c_3^{(k)} & c_4^{(k)} & c_5^{(k)} \end{bmatrix}\begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}. \tag{1.11}$$

Our task is to estimate the parameters $\{c_k\}_{k=0}^5$ that minimize the difference between f_k and f_0, that is,

$$(c_0^{(k)}, c_1^{(k)}, \cdots, c_5^{(k)}) = \text{argmin} \sum_{j \in \mathcal{I}} [f_k(R_k \mathbf{x}_j - \mathbf{r}_k) - f_0(\mathbf{x}_j)]^2,$$

where $\mathcal{I}$ is the index set of pixels in the region of interest, which may be the entire image or part of the image. Many methods can be used to solve this minimization problem, such as the Levenberg-Marquardt iterative nonlinear minimization algorithm S96.

With R_k and $\mathbf{r_k}$, we can compute the sensor position (s_k^x, s_k^y) with $s_k^x, s_k^y \in \{0, 1\}$ and the displacement errors $(\epsilon_k^x, \epsilon_k^y)$ for the frame f_k with respect to f_0. Since $f_k(R_k\mathbf{x} - \mathbf{r}_k) = f_k(R_k(\mathbf{x} - R_k^{-1}\mathbf{r}_k))$, it can be viewed as a translation of f_0 with displacement vector $-R_k^{-1}\mathbf{r}_k$. Our task is to write

$$R_k^{-1}\mathbf{r}_k = \mathbf{u}_k + \frac{1}{2}\begin{bmatrix} s_k^x \\ s_k^y \end{bmatrix} + \frac{1}{2}\begin{bmatrix} \epsilon_k^x \\ \epsilon_k^y \end{bmatrix}. \tag{1.12}$$

Then, $\hat{f}_k(\mathbf{x}) \equiv f_k(R_k(\mathbf{x} - \mathbf{u}_k))$ can be considered as the low-resolution image $g_{s_k^x, s_k^y}$ with displacement errors $(\epsilon_k^x, \epsilon_k^y)$ at sensor position (s_k^x, s_k^y). The algorithm is as follows.

Algorithm 1. $(\hat{f}(\mathbf{x}), s^x, s^y, \epsilon^x, \epsilon^y) \leftarrow (f, f_0)$*: locate the frame f against the reference frame f_0.*

```
1 Compute $[\tilde{r}_1, \tilde{r}_2] = R^{-1}\mathbf{r}$.
2 Let $\mathbf{u} \equiv [\lfloor \tilde{r}_1 + \frac{1}{4} \rfloor, \lfloor \tilde{r}_2 + \frac{1}{4} \rfloor]^\top$, then $[d_1, d_2] \equiv [\tilde{r}_1, \tilde{r}_2] - \mathbf{u}^\top$ has entries in $[-\frac{1}{4}, \frac{3}{4})$.
3 Let $[s^x, s^y] \equiv [\lfloor 2d_1 + \frac{1}{2} \rfloor, \lfloor 2d_2 + \frac{1}{2} \rfloor]$, then $s^x, s^y \in \{0, 1\}$.
4 Let $[\epsilon^x, \epsilon^y] \equiv [2d_1 - s^x, 2d_2 - s^y]$, then $|\epsilon^x|, |\epsilon^y| < \frac{1}{2}$ and (1.12) holds.
5 $\hat{f}(\mathbf{x}) \equiv f(R(\mathbf{x} - \mathbf{u}))$.
```

We use Figure 1.4 to illustrate how the algorithm resolves the displacement in the horizonal direction. It is similar for the vertical direction.

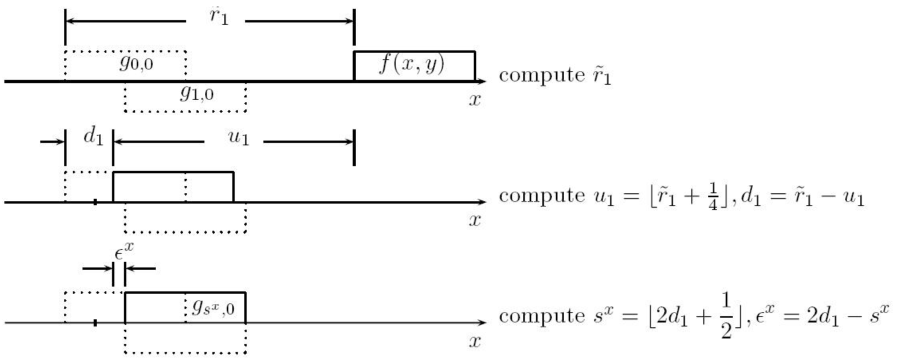

$\tilde{r}_1$
$g_{0,0}$
$g_{1,0}$
$f(x,y)$
compute $\tilde{r}_1$
x
d_1
u_1
compute $u_1 = \lfloor \tilde{r}_1 + \frac{1}{4} \rfloor, d_1 = \tilde{r}_1 - u_1$
x
ϵ^x
$g_{s^x,0}$
compute $s^x = \lfloor 2d_1 + \frac{1}{2} \rfloor, \epsilon^x = 2d_1 - s^x$
x

Figure 1.4. Resolve the horizonal displacement in Algorithm 1. First we compute the total displacement $\tilde{r}_1$. Then write $\tilde{r}_1 = u_1 + d_1$ where $d_1 \in [-1/4, 3/4)$. Then the sensor location $s^x = \lfloor 2d_1 + \frac{1}{2} \rfloor$ and the displacement error $\epsilon^x = 2d_1 - s^x$.

The Video Still Enhancement Algorithm

After passing a frame f through Algorithm 1, we then have the (s^x, s^y)th low-resolution image with displacement error (ϵ^x, ϵ^y), i.e.

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

But in the algorithm for image enhancement (1.10) (see also Figure 1.3), we assume not one, but a complete set of low-resolution images $\{\tilde{g}_{i,j}\}_{i,j=0}^{1}$ at every sensor position. To compensate for the missing low-resolution images, our idea is to generate them by downsampling the current high-resolution approximation h of f_0 with zero displacement error, i.e.

$$\mathbf{g}_{i,j} = \begin{cases} \hat{\mathbf{f}} - D_{i,j}(H - H_{0,0})\mathbf{h}, & (i,j) = (s^x, s^y), \\ D_{i,j}H_{0,0}\mathbf{h}, & (i,j) \neq (s^x, s^y). \end{cases} \tag{1.13}$$

We use an alternate direction approach to obtain the high-resolution image $\mathbf{h}$. In the m-th outer iteration step, we let $\mathbf{g}_{s^x,s^y} = \hat{\mathbf{f}} - D_{s^x,s^y}(H - H_{0,0})\mathbf{h}_m^{(0)}$, where $\mathbf{h}_m^{(0)} = \mathbf{h}_{m-1}$. Then, we iterate $\mathbf{h}_m^{(n)}$ with respect to n as shown in Figure 1.5, which is in fact a modification of Figure 1.3. When it converges, we set $\mathbf{h}_{m+1}^{(0)} = \mathbf{h}_m$. Once we get an update $\mathbf{h}_m$, we will go into the next outer iteration; see Figure 1.6. The complete alternate direction algorithm is as follows.

Algorithm 2. $\mathbf{h} \leftarrow$ *Update* $(\mathbf{h}, \hat{\mathbf{f}}, s^x, s^y, \epsilon^x, \epsilon^y)$*: Update the high resolution image* $\mathbf{h}$ *by a frame* $\hat{\mathbf{f}}$ *with parameters* $(s^x, s^y, \epsilon^x, \epsilon^y)$.

1 Initialize $\mathbf{h}_0 = \mathbf{h}$, and set $m = 0$.

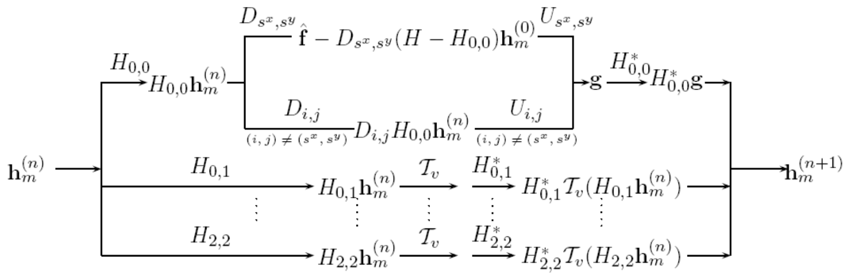

D_{s^x,s^y}
$\hat{\mathbf{f}} - D_{s^x,s^y}(H - H_{0,0})\mathbf{h}_m^{(0)}$
U_{s^x,s^y}
$H_{0,0}$
$H_{0,0}\mathbf{h}_m^{(n)}$
$D_{i,j}$
$(i,j) \neq (s^x,s^y)$
$D_{i,j}H_{0,0}\mathbf{h}_m^{(n)}$
$U_{i,j}$
$(i,j) \neq (s^x,s^y)$
$\mathbf{g}$
$H_{0,0}^*$
$H_{0,0}^*\mathbf{g}$
$\mathbf{h}_m^{(n)}$
$H_{0,1}$
$H_{0,1}\mathbf{h}_m^{(n)}$
$\mathcal{T}_v$
$H_{0,1}^*$
$H_{0,1}^*\mathcal{T}_v(H_{0,1}\mathbf{h}_m^{(n)})$
$\mathbf{h}_m^{(n+1)}$
$H_{2,2}$
$H_{2,2}\mathbf{h}_m^{(n)}$
$\mathcal{T}_v$
$H_{2,2}^*$
$H_{2,2}^*\mathcal{T}_v(H_{2,2}\mathbf{h}_m^{(n)})$

Figure 1.5. Inner iteration of Algorithm 2. Here, we fix the low-resolution image $\mathbf{g}_{s^x,s^y}$, and compensate the missing ones $\mathbf{g}_{i,j}$ by downsampling $H_{0,0}\mathbf{h}_m^{(n)}$. Then we have $\mathbf{g} = \sum_{i,j=0}^{1} U_{i,j}\mathbf{g}_{i,j}$ as in (1.9) with $\mathbf{g}_{i,j}$ substituted by (1.13). We iterate $\mathbf{h}_m^{(n)}$ w.r.t. n until it converges; then we set $\mathbf{h}_m = \mathbf{h}_m^{(n)}$.

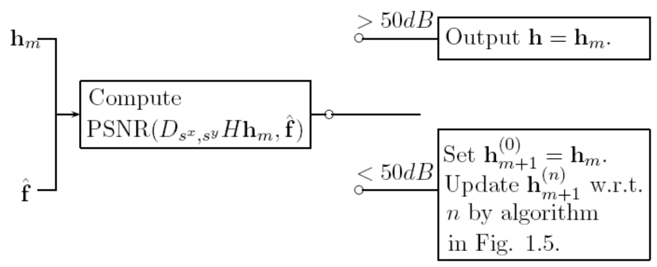

$\mathbf{h}_m$
$\hat{\mathbf{f}}$
Compute
PSNR$(D_{s^x,s^y}H\mathbf{h}_m, \hat{\mathbf{f}})$
$> 50dB$
Output $\mathbf{h} = \mathbf{h}_m$.
$< 50dB$
Set $\mathbf{h}_{m+1}^{(0)} = \mathbf{h}_m$.
Update $\mathbf{h}_{m+1}^{(n)}$ w.r.t.
n by algorithm
in Fig. 1.5.

Figure 1.6. Outer iteration of Algorithm 2. We update the high resolution image $\mathbf{h}$ by a frame $\hat{\mathbf{f}}$ with parameters $(s^x, s^y, \epsilon^x, \epsilon^y)$.

2 If $PSNR(D_{s^x,s^y}H\mathbf{h}_m, \hat{\mathbf{f}}) < 50dB$, set $\mathbf{h}_{\mathbf{m+1}}^{(\mathbf{0})} = \mathbf{h}_m$, and $m = m+1$; otherwise, output $\mathbf{h} = \mathbf{h}_m$, stop.

3 Iterate $\mathbf{h}_m^{(n)}$ w.r.t. n until convergence (see Figure 1.5):

(a) $$\mathbf{g}_{i,j} = \begin{cases} \hat{\mathbf{f}} - D_{s^x,s^y}(H - H_{0,0})\mathbf{h}_m^{(0)}, & (i,j) = (s^x, s^y), \\ D_{i,j}H_{0,0}\mathbf{h}_m^{(n)}, & (i,j) \neq (s^x, s^y). \end{cases}$$

(b) $$\mathbf{g} = \sum_{i,j=0}^{1} U_{i,j}\mathbf{g}_{i,j}.$$

(c) $$\mathbf{h}_m^{(n+1)} = H_{0,0}^*\mathbf{g} + \sum_{\substack{i,j=0 \\ (i,j) \neq (0,0)}}^{2} H_{i,j}^*\mathcal{T}_v(H_{i,j}\mathbf{h}_m^{(n)}).$$

4 Set $\mathbf{h}_m = \mathbf{h}_m^{(n)}$ after converge, and go back to step 2.

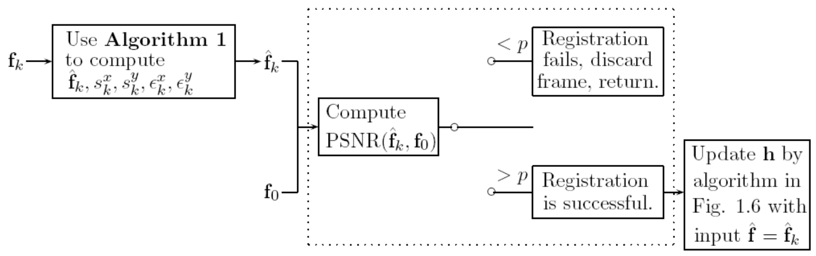

$\mathbf{f}_k$
Use **Algorithm 1** to compute $\hat{\mathbf{f}}_k, s_k^x, s_k^y, \epsilon_k^x, \epsilon_k^y$
$\hat{\mathbf{f}}_k$
$\mathbf{f}_0$
Compute PSNR$(\hat{\mathbf{f}}_k, \mathbf{f}_0)$
$< p$
Registration fails, discard frame, return.
$> p$
Registration is successful.
Update $\mathbf{h}$ by algorithm in Fig. 1.6 with input $\hat{\mathbf{f}} = \hat{\mathbf{f}}_k$

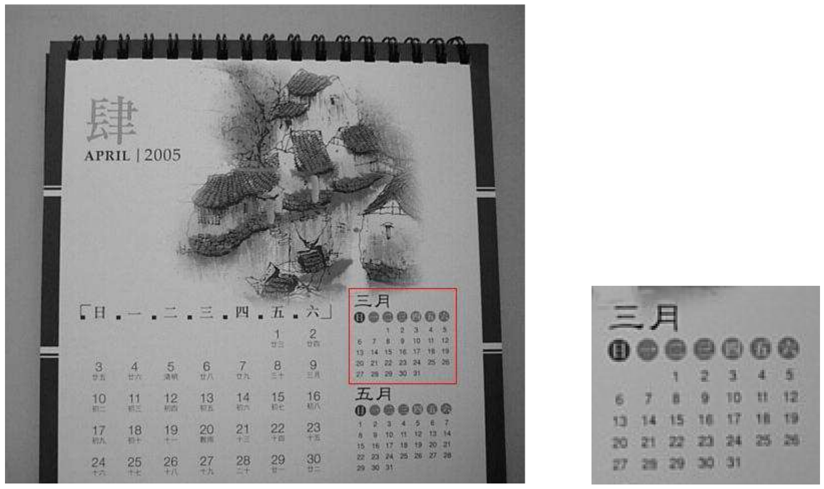
Figure 1.7. Resolution enhancement for video clips (in the dotted-lined box it is used to determine if $\hat{\mathbf{f}}_k$ is good enough to update $\mathbf{h}$).

In Figure 1.7, we give the algorithm for the video still enhancement. Given a reference frame $\mathbf{f}_0$, we use a sequence of $2K$ frames $\{\mathbf{f}_k\}_{k=-K}^{K}$ that are taken just before and after the reference frame $\mathbf{f}_0$. The step in the dotted-lined box is to determine if the shifted frame $\hat{\mathbf{f}}_k$ is close enough to $\mathbf{f}_0$ or else we discard the frame. In the experiments, we set $p = 25dB$. Initially, we estimate $\mathbf{h}$ by bilinear interpolation on $\mathbf{f}_0$, and then use the new information from good frames to update $\mathbf{h}$. The advantage of our algorithm is that based on the rule shown in the dotted-lined box it only chooses the good candidate frames to enhance the resolution, and there is no need to determine the number of frames to be used in advance.

3. Video Enhancement Algorithm

Since video streams are made up of frames, and we can now improve the resolution of each frame in a video stream, we can compose these high-resolution frames together to generate a higher resolution video of the given video stream. More precisely, we can apply our algorithm in Figure 1.7 to the frames $\{f_k\}_{k=-K}^{K}$ to enhance f_0, and then apply the algorithm again to frames $\{f_k\}_{k=-K+1}^{K+1}$ to enhance f_1, etc. Then, by combining the enhanced frames, we obtain a high-resolution video.

In this section, we test this idea for a video clip which is filmed by us by moving our camera over a calendar. The video clip is in .avi format with size 520×480, and can be downloaded at VIDEO. We first try the video still enhancement algorithm in CSX07 to enhance the resolution of a frame in the video. In the seven seconds of the video, we choose the 60th frame f_{60} as our reference frame, see Figure 1.8 (a). In Figure 1.8(b), we show a part of the reference frame f_{60} (the area enclosed by the box in Figure 1.8 (a)) that we try to improve the resolution on.



(a) (b)

Figure 1.8. (a) The reference frame f_{60}, and (b) a part of f_{60} .

We let $K = 10$, that is, we use the 50th to the 70th frames to improve the resolution of f_{60}. The alignment parameters for this clip are listed in Table 1.1, which shows that frames f_{52} and f_{54} are discarded. Figure 1.9(a) gives the first guess of the high-resolution image of f_{60} by the bilinear interpolation. The result from our algorithm (i.e. Figure 1.7) is shown in Figure 1.9(b). Clearly the calendar by our method is much clearer than that by the bilinear interpolation. Moreover some numbers, such as "16" and "18", which are clearly discernible now, are very difficult to read from the video clip or just by bilinear interpolation.

The results clearly show that the video still enhancement algorithm (Figure 1.7) is working. Next we extend it to video enhancement. Our aim is to obtain a high-resolution video for the image in Figure 1.8(b). We will use the video still enhancement algorithm (Figure 1.7) repeatedly to enhance all frames in $\{f_k\}_{k=40}^{63}$. More precisely, frames $\{f_k\}_{k=\ell-10}^{\ell+10}$ will be used to improve the resolution of frames f_ℓ, with $40 \leq \ell \leq 63$. The enhanced stills are put back together to get a higher-resolution video with 24 frames. The original clip and the resulting clips are given in VIDEO. Because the new one has higher resolution, it is 4 times in size and is much clearer.

Table 1.1. Alignment Results from Our Algorithm

Frame Index	(s^x, s^y)	(ϵ^x, ϵ^y)	$f_0(x) \approx f(R\mathbf{x} + \mathbf{r})$
61	(1,0)	(0.119, 0.036)	Yes
59	(1,0)	(0.246,-0.066)	Yes
62	(0,0)	(0.368, 0.186)	Yes
58	(0,0)	(0.272,-0.126)	Yes
63	(1,0)	(-0.139, 0.086)	Yes
57	(1,0)	(0.334,-0.214)	Yes
64	(1,0)	(0.323,-0.194)	Yes
56	(0,0)	(-0.134,-0.381)	Yes
65	(0,0)	(-0.349, 0.164)	Yes
55	(0,1)	(0.454, 0.448)	Yes
66	(0,0)	(0.421, 0.181)	Yes
54	$\cdots$	$\cdots$	No
67	(1,0)	(-0.219, 0.236)	Yes
53	(0,1)	(-0.323, 0.323)	Yes
68	(1,0)	(0.313, 0.232)	Yes
52	$\cdots$	$\cdots$	No
69	(0,0)	(0.096, 0.204)	Yes
51	(0,0)	(0.292,-0.500)	Yes
70	(0,0)	(0.485, 0.186)	Yes
50	(0,1)	(0.062, 0.301)	Yes

4. Conclusion

In this paper, we give a short survey of the framelet algorithm proposed in CSX07 for high-resolution still enhancement from video clips. We then extend it to video enhancement. Simulation results show that our framelet algorithm can reveal information that is not discernible in the original video clips or by simple interpolation of any particular frame in the video.

By modification of the motion estimation equation (1.11), our framelet algorithm can also be extended to more complicated motions. So far, we have not yet make use of the sparsity of the tight-frame coefficients across frames. How to make use of it is an interesting topic for our future work.

References

Mather, P.. (2004). *Computer Processing of Remotely-Sensed Images: An Introduction.* 3rd Edition, John Wiley & Sons, Chichester (UK).

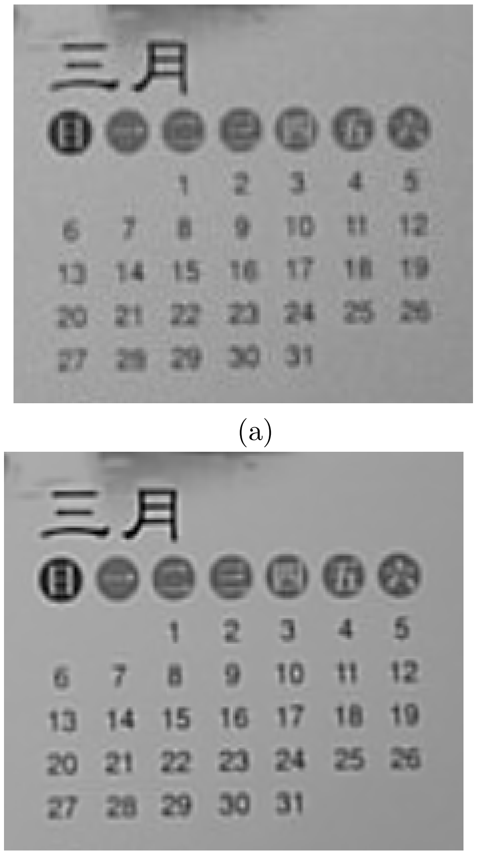


Figure 1.9. Reconstructed high-resolution images: (a) using bilinear interpolation, (b) using our algorithm in Figure 1.7.

Lillesand M. T., Kiefer W. R., and Chipman W. J.. (2000). *Remote Sensing and Image Interpretation*. 4th edition, John Wiley & Sons.

Capel D. and Zisserman A.. (2003). "Computer vision applied to super-resolution," *IEEE Signal Processing Magazine*, 20:72–86.

KIM S. P. and SU W.-Y.. (1993). "Recursive high-resolution reconstruction of blurred multiframe images", *IEEE Transactions on Image Processing*, 2:534–539.

Nakazawa Y., Komatsu T., and Saito T.. (1995). "High-resolution image acquisition based on temporal integration with hierarchical estimation of image warping", *Proceedings of IEEE International Conference on Image Processing*, Washington, DC, pp. 244–247.

Schultz R. R. and Stevenson R. L.. (1996). "Extraction of high-resolution frames from video sequences", *IEEE Transactions on Image Processing*, 5:996–1011.

Eren P. E., Sezan M. I., and Tekalp A. M.. (1997). "Robust, object-based high-resolution image reconstruction from low-resolution video", *IEEE Transactions on Image Processing*, 6:1446–1451.

Yang Q. and Parvin B.. (2003). "High-Resolution reconstruction of sparse data from dense low-resolution spatio-temporal data", *IEEE Transactions on Image Processing*, 12:671–677.

Mateos J., Molina R., and Katsaggelos A. K.. (2003). "Bayesian high resolution image reconstruction with incomplete multisensor low resolution systems", *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Hong Kong, pp. 705–708.

Chan H. R., Riemenschneider S. D., Shen L. X., and Shen Z. W.. (2004). "Tight frame: The efficient way for highresolution image reconstruction", *Applied and Computational Harmonic Analysis*, 17:91–115.

Chan H. R., Riemenschneider S. D., Shen L. X., and Shen Z. W.. (2004). "Highresolution image reconstruction with displacement errors: A framelet approach", *International Journal of Imaging Systems and Technology*, 14:91–104.

Chan H. R., Shen Z. W., and Xia T.. (2007). "A Framelet Algorithm for Enhancing Video Stills", *Applied and Computational Harmonic Analysis*, 23:153-170.

Bose N. and Boo K.. (1998). "Highresolution image reconstruction with multisensors", *International Journal of Imaging Systems and Technology*, 9:294–304.

Ron A. and Shen Z. W.. (1997). "Affine system in $L^2(R^d)$: the analysis of the analysis operator", *Journal of Functional Analysis*, 148:408–447.

Donoho D.. (1995). "De-noising by soft-thresholding", *IEEE Transactions on Information Theory*, 41:613–627.

Szeliski R.. (1996). "Video mosaics for virtual environments", *IEEE Computer Graphics and Applications*, Mar. pp. 22–30.

Video clip and full results at `http://www.math.cuhk.edu.hk/~rchan/paper/cdw`.

Cain S., Hardie R. C., and Armstrong E. E.. (1996). "Restoration of aliased video sequences via a maximum-likelihood approach", *Infrared Information Symposium (IRIS) on Passive Sensors*, 1:377-390.

Hardie R. C., Barnard K. J., Bognar J. G., Armstrong E. E., and Watson E. A.. (1998). "High-resolution image reconstruction from a sequence of rotated and translated frames and its application to an infrared imaging system", *Optical Engineering*, 37:247-260.

Cai J.-F., Chan R. H., and Shen Z.W.. (2008). "A Framelet-Based Image Inpainting Algorithm", *Applied and Computational Harmonic Analysis*, 24:131-149.