Computational Cost Reduction of Improved Super-Resolution Method Using Overlapped Block Matching

Yasuo TAKEHISA† (Member), Kiyoshi TANAKA†† (Member)
† Nagano Prefecture General Industrial Technology Center
†† Faculty of Engineering, Shinshu University

Summary The improved super-resolution method that achieves dense motion estimation (DME) using overlapped block matching (OBM) remarkably improves the quality of reconstructed images for a given video sequence. However, this method has a drawback to increase the computational cost almost linearly to the number of overlapped blocks because DME using OBM allocates multiple motion vectors to a local region in the image restoration process. To solve this problem, in this paper we propose a method to reduce computational cost of the improved super-resolution method by considering the statistics of motion vectors obtained by DME using OBM. This method can reduce the entire computational cost up to 29.9 ∼ 49.1% depending on a given video sequence while completely maintaining the original performance of the improved super-resolution method using OBM. Also, we try to further reduce computational cost by relaxing the complete original performance preservation requirement. With this additional attempt, we can further reduce computational cost up to 16.9 ∼ 20.8% without serious deterioration of the quality of reconstructed images.

Key words: Super-resolution, Computational cost reduction, Dense Motion Estimation (DME), Overlapped Block Matching (OBM), Motion vector

1. Introduction

Super-resolution 1)−4) is a technique which tries to generate (reconstruct) high-resolution images by making use of some useful information contained in low-resolution captured images. In the research field of super-resolution, the effectiveness of dense motion estimation (DME) that allocates motion vector pixel by pixel has been reported. Hong et al. 5) proposed to use DME realized by hierarchical block matching in super-resolution. Although Hong's DME is useful for high quality observed images, the performance noticeably deteriorates for low quality images that are greatly required to improve resolution and image quality. On the other hand, Takehisa et al. 6) proposed a method to achieve DME virtually in super-resolution using overlapped block matching (OBM).

While this method does not improve the accuracy of motion estimation, it apparently increases the number of reference frames by allocating multiple motion vectors to a small region, which remarkably improves the performance of super-resolution. However, this method has a drawback to increase the computational cost almost linearly to the number of overlapped blocks.

To solve this problem, in this paper we propose a method to reduce computational cost of the improved super-resolution method 6) by considering the statistics of motion vectors obtained by DME using OBM. We locally observe the statistics of motion vectors allocated to a small cell consisting of 2 × 2 pixels, and reduce computational cost by conjoining the calculation for same motion vectors in optimization process. This method can reduce the entire com-
computational cost up to 29.9 ~ 49.1% depending on
a given video sequence while completely maintain-
ing the original performance of the improved super-
resolution method using OBM. Also, we try to fur-
ther reduce computational cost by relaxing the com-
plete original performance preservation requirement.
With this additional attempt, we can further reduce
computational cost up to 16.9 ~ 20.8% without seri-
sous deterioration of the quality of reconstructed im-
ages.

2. Principle of Super-Resolution

2.1 Video Sequence Observation Model

The improved super-resolution method is based
on Schultz and Stevenson’s method, in which they
describe the process that produces observed low-
resolution frames from a high-resolution frame with
the following equations.

\[
\begin{align*}
\vdots \\
y^{(k-1)} &= DF^{(k-1),k}z^{(k)} + n^{(k-1),k} \\
y^{(k)} &= Dz^{(k)} \\
y^{(k+1)} &= DF^{(k+1),k}z^{(k)} + n^{(k+1),k} \\
\vdots \\
\end{align*}
\]

- \( z^{(k)} \): k-th high-resolution image vector
- \( y^{(l)} \): Observed l-th low-resolution image vector
- \( F^{(l,k)} \): Motion estimation matrix between \( z^{(k)} \)
  and \( z^{(l)} \)
- \( n^{(l,k)} \): Noise vector produced by \( F^{(l,k)} \)
- \( D \): Downsampling matrix

These equations mean that neighbor high-
resolution frames \( z^{(l)} \) \((l = \ldots, k-1, k+1, \ldots)\) are gen-
erated by applying motion estimation matrix \( F^{(l,k)} \)
to k-th high-resolution frame \( z^{(k)} \), and its downsam-
pled ones \( y^{(l)} \) are observed with noise vector \( n^{(l,k)} \).
\( y^{(k)} \) is obtained by just downsampling \( z^{(k)} \) because of
no motion estimation.

2.2 Estimation of High-Resolution Images

Because the problem estimating a high-resolution
frame \( z^{(k)} \) from observed low-resolution frames \( \{y^{(l)}\} \)
\( (l = \ldots, k-1, k, k+1, \ldots) \) is ill-posed, we cannot
specify the solution uniquely. With this reason, we
consider a solution \( \hat{z}^{(k)} \) that maximizes the posterior
probability \( Pr(z^{(k)} | \{y^{(l)}\}) \) as the best solution based
on MAP estimation.

\[
\hat{z}^{(k)} = \arg \max_{z^{(k)}} Pr(z^{(k)} | \{y^{(l)}\}) \tag{2}
\]

We can rewrite the solution \( \hat{z}^{(k)} \) as a summation of
conditional and prior probabilities by applying Bayes’
theorem to the posterior probability \( Pr(z^{(k)} | \{y^{(l)}\}) \)
in Eq. (2) and taking logarithm entirely.

\[
\hat{z}^{(k)} = \arg \max_{\tilde{z}^{(k)}} \left\{ \sum_{l} \log Pr(y^{(l)} | \tilde{z}^{(k)}) + \log Pr(\tilde{z}^{(k)}) \right\} \tag{3}
\]

The first term in Eq. (3) is a conditional probability
that the low-resolution frame becomes \( y^{(l)} \) when a
high-resolution frame is \( z^{(k)} \). Although this proba-
bility distribution should be equivalent to the one for
noise \( n^{(l,k)} \), we cannot estimate the accurate shape.
Here we assume the normal distribution with average
0 and variance \( \sigma^{(l,k)} \). The second term in Eq. (3)
is a prior probability of high-resolution frame \( z^{(k)} \).
This follows the Gibbs probability distribution by
assuming that the model of \( z^{(k)} \) is Markov random
field. Also, we consider the locality of pixels in the
frame (smooth edge will be more probable than sharp
one) by using Huber function. The balance between
first and second terms are properly adjusted. Finally,
Eq. (3) is solved by a certain optimization method
with the following constraint:

\[
y^{(k)} = Dz^{(k)} \tag{4}
\]

3. DME Using OBM in Super-Resolution

3.1 Concept of DME using OBM

In the improved super-resolution method, OBM
is used to extract much more useful information for
super-resolution from a limited number of reference
frames than Schultz and Stevenson’s super-resolution
method. Here we illustrate how to use OBM to re-
alize DME virtually in the case that two blocks are
overlapped. We divide the same frame into blocks
two times by differentiating offset positions as shown
in Fig. 1, and carry out motion estimation for all
blocks obtained. In this figure, we divided the same
frame in two ways: normal image division (Fig. 1(a))
and shifted image division (Fig. 1(b)). Since each di-
vision has different offset position, the size of sub-
divided regions in the target frame becomes small
(1/4 in this case) by overlapping both divisions as
shown in Fig. 1(c). This increases the chance to allo-
cate different motion vectors for each sub-divided re-
region by overlapping. We call the blocks obtained by normal image division, the ones by shifted image division and the overlapped regions by both divisions as “basic blocks”, “shifted blocks” and “virtual blocks”, respectively.

3.2 Effect from DME using OBM

Now, we explain the effect from multiple motion vectors estimation by OBM in detail. Let us consider a pair of basic and shifted blocks, \( B_A \) and \( B_B \), in Fig. 2, where both blocks have common region in them. However, since \( B_A \) and \( B_B \) may have different pixel patterns, the possibility that different motion vector is allocated to each block becomes high. In Fig. 2(a), a motion vector \( v_A \) is allocated to \( B_A \) and a different one \( v_B \) is allocated to \( B_B \), i.e. \( v_A \neq v_B \). Note here that as shown in Fig. 2(b) hatched areas in \( B'_A \) and \( B'_B \) are obviously different regions, but they refer a same virtual block. In other words, the same information on the reference frame is twice provided to different regions on the target frame. In this sense, we can increment block numbers two times for virtual blocks satisfying \( v_A \neq v_B \). If we find \( n_{\text{dif}} \) virtual blocks satisfying \( v_A \neq v_B \) among all \( n_{\text{vir}} \) virtual blocks in a reference frame, we can consider that the reference frame contains totally \( n_{\text{vir}}^+ = n_{\text{vir}} + n_{\text{dif}} \) reference blocks. Here we can define the virtual reference frame numbers by the following equation.

\[
F^+ = \frac{n_{\text{vir}}^+}{n_{\text{vir}}} \tag{5}
\]

If \( n_{\text{dif}} > 0, n_{\text{vir}}^+ > n_{\text{vir}} \) and thus \( F^+ > 1 \). This is apparently equivalent to increase the number of reference frames, and thus we can expect the performance improvement in super-resolution. On the other hand, if we focus on the size of virtual block, the size becomes half in both vertical and horizontal directions by overlapping basic and shifted blocks like the example of Fig. 1(c). In this way, OBM works to reduce the size of virtual block as well.

The increase of overlapped block numbers in OBM by adding shifted blocks with different offset positions urges (i) the increase of virtual reference frame numbers, and (ii) the size reduction of virtual block, simultaneously. We show an example of OBM in the case that the block size for image division is \( d \times d = 4 \times 4 \) pixels. In the case that the number of overlapped blocks is \( N = 2 \) as shown in Fig. 3(a), the size of the virtual block becomes \( m \times m = 2 \times 2 \) pixels, to which we can allocate \( N = 2 \) motion vectors. On the other hand, in the case that the number of overlapped blocks is \( N = 4 \) as shown in Fig. 3(b), the size of the virtual block becomes \( m \times m = 1 \times 1 \) pixel, to which we can allocate \( N = 4 \) motion vectors. In the latter case we can consider that pixel-level DME is virtually realized by OBM. In this way, the improved super-resolution method gives much motion information to a small region (virtual block), which achieves stable (robust) performance in super-resolution.\(^6\)
4. Proposed Method

4.1 Problem

First of all, we explain the procedure of image restoration in Schultz and Stevenson’s super-resolution method⁷ using Fig. 4, in which we illustrate the operation between k-th and (k + 1)-th frames. First we approximate (k + 1)-th reference frame \( F^{(k+1)} \) by using motion vectors found in the k-th target frame for each block divided in the (k + 1)-th reference frame. Here, \( \hat{z} \) denotes the estimated k-th high-resolution frame. Next, the generated reference frame is downsampled to measure SSD (sum of squared differences) error \( \varepsilon^{(k+1)} \) between \( DF^{(k+1)} \hat{z}^{(k)} \) and the observed reference frame \( y^{(k+1)} \). Then, the target frame \( \hat{z}^{(k)} \) is updated such that the accumulated error \( \varepsilon_{all} \) for all reference frames is minimized under the constraint that \( Dz^{(k)} = y^{(k)} \). This procedure is iterated until the pre-specified condition on error is satisfied.

On the other hand, in the improved super-resolution method⁶, we divide the same reference frame \( N \) times by differentiating the offset positions, and conduct \( N \) times of motion estimation. Accordingly, \( N \) motion vectors \( v_i \) (\( i = 1, 2, \cdots, N \)) are allocated to each virtual block formed by overlapping \( N \) kinds of image division. Table 1 shows the number of motion vectors (= number of overlapped blocks) \( N \) and the corresponding size of virtual block \( (m \times m \) pixels). Thus, we must generate \( N \) approximated frames \( F^{(k+1)}z^{(k)} \) (\( i = 1, 2, \cdots, N \)) for a single reference frame by using \( v_i \) (\( i = 1, 2, \cdots, N \)), and measure SSD error \( N \) times \( \varepsilon_i^{(k+1)} \) (\( i = 1, 2, \cdots, N \)) between each downsampled generated reference frame \( DF_1^{(k+1)} \hat{z}^{(k)} \) (\( i = 1, 2, \cdots, N \)) and the observed frame \( y^{(k+1)} \). We show an example for \( N = 2 \) in Fig. 5, in which two SSD errors \( \varepsilon_1^{(k+1)} \) and \( \varepsilon_2^{(k+1)} \) are separately measured between one downsampled reference frame \( DF_1^{(k+1)} \hat{z}^{(k)} \) for basic image division and the observed reference frame \( y^{(k+1)} \), and another downsampled reference frame \( DF_2^{(k+1)} \hat{z}^{(k)} \) for shifted image division and \( y^{(k+1)} \). Then all the errors are accumulated for all reference frames. This procedure should be done every time when the target frame \( \hat{z}^{(k)} \) is updated, which seriously enlarges the computational cost required for image restoration as the number of overlapped blocks \( N \) increases.

4.2 Method to Reduce Computational Cost

Now, we focus on the statistics of \( N \) motion vectors \( v_i \) (\( i = 1, 2, \cdots, N \)) allocated to each virtual block. We can expect high correlation among some same motion vectors appeared in the set of \( \{ v_i \} \).

![Fig. 4](image-url) Procedure of image restoration in Schultz and Stevenson’s super-resolution method⁷

![Fig. 5](image-url) Procedure of image restoration in the improved super-resolution method⁶ (\( N = 2 \))
is quite high. The proposed method reduces computational cost by conjoining the calculation for same motion vectors in optimization process.

Here let us observe the statistics of motion vectors locally for a small cell consisting of \(2 \times 2\) pixels because SSD error is calculated between pixels after downsampling by averaging \(2 \times 2\) pixels. Fig. 6 shows the calculation of SSD error in a cell, which can be considered a microscopic illustration of Fig. 4. \(v_{11}, v_{12}, v_{21}\) and \(v_{22}\) are motion vectors for each pixel in the cell. \(4\) pixels, \(q_{11}, q_{12}, q_{21}\) and \(q_{22}\) on the \(k\)-th target frame specified by \(v_{11}, v_{12}, v_{21}\) and \(v_{22}\), are copied to approximate the \((k+1)\)-th reference frame. Then, the cell is downsampled to a single pixel \(p\) by averaging \(4\) copied pixels, and SSD error between \(p\) and \(r\) that is the corresponding pixel on the \((k+1)\)-th observed frame is calculated. Note that the "cell" is different from virtual blocks and/or blocks used for motion estimation.

Next, we explain the way of conjoining the calculation of SSD error for same motion vectors using Fig. 7, where the number of motion vectors allocated to a single pixel is \(N = 4\). Since \(N = 4\), each pixel has \(4\) motion vectors \(v_i\) \((i = 1, 2, 3, 4)\). Thus, we have \(4\) sets of motion vectors for a cell, \(V_{2 \times 2}^{(i)} = \{v_{11}^{(i)}, v_{12}^{(i)}, v_{21}^{(i)}, v_{22}^{(i)}\}\) \((i = 1, 2, 3, 4)\). In this example, \(V_{2 \times 2}^{(1)} = V_{2 \times 2}^{(2)} = V_{2 \times 2}^{(3)} \neq V_{2 \times 2}^{(4)}\). Thus, the number of different sets of motion vectors in the cell is \(\eta_s = 2\), where \(s = 1, 2, \ldots, n_{\text{cell}}\) is the index of a cell and \(n_{\text{cell}}\) is the total number of cells in a frame. Here we denote different sets of motion vectors for the \(s\)-th cell \(c_s\) as \(V_s^{(j)}\) \((j = 1, 2, \ldots, \eta_s)\). Also, we denote the number of motion vector sets belonging to each \(V_s^{(j)}\) \((j = 1, 2, \ldots, \eta_s)\) as \(\zeta_s^{(j)}\) \((j = 1, 2, \ldots, \eta_s)\). In this example, \(\zeta_s^{(1)} = 3\) and \(\zeta_s^{(2)} = 1\) \((\sum_{j=1}^{\eta_s} \zeta_s^{(j)} = N)\). For a same set of motion vectors \(V_s^{(j)}\), we can conjoin the calculation of (i) approximation of cell using motion vectors, (ii) downsampling pixels in the cell, and (iii) subtraction and square operations to measure SSD error as shown in Fig. 8, which gives pseudo codes of a part of the optimization process. In the proposed method, the most inner loop is truncated by \(\eta_s\) times, while the improved super-resolution method \(6\) constantly iterates this loop \(N\) times. (Note that we sometimes call Ref.6) "conventional method” with the reference number to distinguish it from the proposed method in the following.)

The truncation of the most inner loop significantly contributes to remove redundant calculations in the optimization process of super-resolution.

Summarizing, the conventional method \(6\) needs constantly
\[
C = N \times n_{\text{cell}}
\]
calculations for reference frame approximation,
Paper: Computational Cost Reduction of Improved Super-Resolution Method

**Fig. 8** A part of pseudo codes in the optimization process

```
begin
  split all approximated reference frames into
  cells (2 × 2 pixels)
  set \( e_{\text{all}} = 0 \)
  for each reference frame \( F_i \) (\( i = 1, 2, \ldots \))
    for each cell \( c_s (s = 1, 2, \ldots, n_{\text{cell}}) \)
      for each different set of motion vectors \( V_s^{(j)} \) (\( j = 1, 2, \ldots, \eta_s \))
        (i) approximate \( \hat{c}_s^{(j)} \) by copying
            \( q^{(j)}_{11}, q^{(j)}_{12}, q^{(j)}_{21}, q^{(j)}_{22} \) and \( q^{(j)}_{33} \) based on \( V_s^{(j)} \).
        (ii) downsample \( \hat{c}_s^{(j)} \) to \( p_s^{(j)} \) by
            \( (q^{(j)}_{11} + q^{(j)}_{12} + q^{(j)}_{21} + q^{(j)}_{22})/4 \).
        (iii) calculate and accumulate SSD error by \( e_{\text{all}} = + \zeta_s \times (p_s - r_s)^2 \)
      done
  done
end
```

**Fig. 9** Distribution of the number of different sets of motion vectors in 2 \( \times \) 2 cell downsampling and measurement of SSD error, while the proposed method does

\[
C' = \sum_{s=1}^{n_{\text{cell}}} \eta_s
\]

(7)
calculations to accomplish same operations. Thus, we can expect to reduce computational cost accordingly to the distribution of \( \{ \eta_s \} \) for a given video sequence.

**Fig. 9** shows the distribution of the number of different sets of motion vectors \( \eta_s \) among \( N \) motion vectors in all reference frames when we realize pixel-level DME (the size of virtual block is \( m \times m = 1 \times 1 \) pixel and the number of overlapped blocks is \( N = 16 \)). We can see from this figure that the distribution of \( \{ \eta_s \} \) is concentrated on the area of small numbers. That is, only a limited number of different motion vectors are appeared among \( N \) sets of motion vectors and used in the optimization process. Since the condition \( \eta_s \ll N \) is satisfied in most cases, we can expect significant reduction of the computational cost of the improved super-resolution method using OBM\(^6\) while completely maintaining the original performance of the method. We call this method “Performance Preserving (PPR) method”.

5. Reduction Effects

We examine the effect of the proposed method (PPR method) to reduce computational cost in the improved super-resolution method\(^6\). As benchmark moving pictures, we used “Mobile Calendar” (352 \( \times \) 288 pixels), “Driving” (352 \( \times \) 240 pixels) and “Foreman” (352 \( \times \) 288 pixels) as original high-resolution images, and their downsampled ones (averaging 4 pixels into one pixel) as observed low-resolution images. As the motion estimation method, in this work we adopt a simple block matching method that calculates the squared error between blocks. We tried to reconstruct the 10-th frame of all observed images using a total of 7 reference frames for “Mobile Calendar” and “Driving”, and 5 reference frames for “Foreman” locating forward and backward of this frame to compare the actual computational time by the conventional\(^6\) and the proposed methods. In our experiments, we use PC implemented a Pentium IV CPU (2.4GHz) and 1.0GB RAM.

We show the results in **Fig. 10**, in which we plot the total computational cost including motion estimation and optimization. As we expected, we can remarkably reduce computational cost by the proposed method comparing with the conventional method\(^6\). The reduction ratio becomes large as we increase the number of overlapped blocks \( N \). For example, in the case of “Mobile Calendar”, we can achieve the reduction ratio of 93.9\%, 73.3\%, 52.1\%, 34.2\% and 29.9\% (\(-6.1\%, -26.7\%, -47.9\%, -65.8\% \) and \(-70.1\% \)) for \( N = 1, 2, 4, 8 \) and 16, respectively.

Especially in the case of pixel-level DME (\( m \times m = 1 \times 1, N = 16 \)), we can reduce up to 29.9\% (\(-70.1\%) \) for “Mobile Calendar”, 43.2\% (\(-56.8\%) \) for “Foreman” and 49.1\% (\(-50.9\%) \) for “Driving”, respectively. We can see that the reduction ratio follows almost in accordance with the distribution of \( \{ \eta_s \} \) shown
Finally, we stress that the proposed method completely maintains the performance of the improved super-resolution method\(^6\). In other words, no PSNR deterioration is caused by this reduction method.

6. Attempt for Further Reduction

In this section, we try to further reduce computational cost by relaxing the requirement that completely maintains the original performance of the improved super-resolution method\(^6\). Unlike PPR method explained in Section 4, here we permit to introduce approximation in the image restoration process. In PPR method, all \(\eta_s\) (where \(1 \leq \eta_s \leq N\)) sets of motion vectors \(\{V^*_s(j)\}\) \((j = 1, 2, \ldots, \eta_s)\) are utilized for each cell \(c_s\) consisting of \(2 \times 2\) pixels. However, here we set an upper bound number for different sets of motion vectors \(N'(<N)\), and restrict different sets of motion vectors used for image restoration within \(N'\). Namely, if \(\eta_s > N'\) for a cell, remaining \(\eta_s - N'\) kinds of \(\{V^*_s(j)\}\) are not utilized, which causes approximation in the optimization process of the improved super-resolution method.

As a criterion to truncate \(\eta_s - N'\) kinds of different sets of motion vectors, we consider the frequency \(\zeta_s(j)\) for each \(V^*_s(j)\) and use only \(N'\) kinds of \(\{V^*_s(j)\}\) having \(N'\) largest frequencies. In this method, we simply do not use \(\eta_s - N'\) kinds of different sets of motion vectors for the operations (i) \sim (iii) in the algorithm shown in Fig. 8. Since \(N'\) is the upper bound, obviously we only use \(\eta_s\) sets of motion vectors if \(\eta_s \leq N'\). Thus the iteration number of the most inner loop is further reduced comparing to the PPR method. We call this method “Performance Non-Preserving (PNP) method”.

The results obtained by PNP method are shown in Table 3, in which PSNR measured for reconstructed image and the actual computational time are listed for \(N' = \{8, 4, 2, 1\}\) only in the case of \(N = 16\) (pixel-level DME). We also include the results by PPR method and the conventional method\(^6\) in the same table for comparison. From the results, we can see that the computational cost can be further reduced as we decrease \(N'\) in PNP method. The reduction ratio depends on a given video sequence with different tendency to the case of PPR method.

Regarding the quality of reconstructed image, ap-
Table 2: Details in the total computational cost

<table>
<thead>
<tr>
<th>N (m × m)</th>
<th>Mobile Calendar</th>
<th>Driving</th>
<th>Foreman</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ref. 6) Proposed Reduction</td>
<td>Ref. 6) Proposed Reduction</td>
<td>Ref. 6) Proposed Reduction</td>
</tr>
<tr>
<td></td>
<td>method ratio</td>
<td>method ratio</td>
<td>method ratio</td>
</tr>
<tr>
<td></td>
<td>[sec] [sec] [%]</td>
<td>[sec] [sec] [%]</td>
<td>[sec] [sec] [%]</td>
</tr>
<tr>
<td>1</td>
<td>16 × 16</td>
<td>motion estimation</td>
<td>1.4 1.4 100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>optimization</td>
<td>31.6 29.6 93.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>total</td>
<td>33 31 93.9</td>
</tr>
<tr>
<td>2</td>
<td>8 × 8</td>
<td>motion estimation</td>
<td>2.92 2.92 100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>optimization</td>
<td>43.9 31.4 71.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>total</td>
<td>46.8 34.3 73.3</td>
</tr>
<tr>
<td>4</td>
<td>4 × 4</td>
<td>motion estimation</td>
<td>5.73 5.9 103</td>
</tr>
<tr>
<td></td>
<td></td>
<td>optimization</td>
<td>72.4 34.8 48.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>total</td>
<td>78.1 40.7 52.1</td>
</tr>
<tr>
<td>8</td>
<td>2 × 2</td>
<td>motion estimation</td>
<td>11.6 11.9 102.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>optimization</td>
<td>183.3 54.7 29.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>total</td>
<td>194.9 66.6 34.2</td>
</tr>
<tr>
<td>16</td>
<td>1 × 1</td>
<td>motion estimation</td>
<td>23.3 23.7 101.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>optimization</td>
<td>395.3 54.7 29.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>total</td>
<td>418.6 66.6 34.2</td>
</tr>
</tbody>
</table>

Table 3: Further computational cost reduction and PSNR achieved

<table>
<thead>
<tr>
<th></th>
<th>Driving</th>
<th>Calendar</th>
<th>Foreman</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ref. 6)</td>
<td>30.741</td>
<td>567.862</td>
<td>26.556</td>
</tr>
<tr>
<td>PPR method</td>
<td>30.741</td>
<td>279.064</td>
<td>26.556</td>
</tr>
<tr>
<td>N' = 8</td>
<td>30.695</td>
<td>219.503</td>
<td>26.597</td>
</tr>
<tr>
<td></td>
<td>(-0.046)</td>
<td>(-38.654%)</td>
<td>(+0.041)</td>
</tr>
<tr>
<td>N' = 4</td>
<td>30.677</td>
<td>166.246</td>
<td>26.605</td>
</tr>
<tr>
<td></td>
<td>(-0.064)</td>
<td>(-29.276%)</td>
<td>(+0.049)</td>
</tr>
<tr>
<td>PNP method</td>
<td>30.587</td>
<td>103.555</td>
<td>26.494</td>
</tr>
<tr>
<td>N' = 2</td>
<td>30.311</td>
<td>84.61</td>
<td>26.659</td>
</tr>
<tr>
<td></td>
<td>(-0.430)</td>
<td>(-14.9%)</td>
<td>(-0.497)</td>
</tr>
</tbody>
</table>

proximation in the image restoration process causes slight variation of PSNR values. In case of “Foreman” we can slightly improve PSNR by PNP method even when we reduce N' until 1. In case of “Mobile Calendar”, we can slightly improve PSNR until N' = 4, but lose PSNR by further decrease of N'. On the other hand, in case of “Driving”, we cannot see PSNR improvement by PNP method, and PSNR gradually deteriorates along with the decrease of N'. The reason why the positive effects (slight PSNR improvement) are observed for some video instances is that truncation of different sets of motion vectors works to reject unreliable sets of motion vectors that do not contribute to reconstruct high-resolution image. This also means that the propriety to use the criterion considering frequency for truncation is confirmed.

As a total conclusion, we can reduce the upper bound number N' until 2 if we accept slight PSNR deterioration on the reconstructed image. When we use N' = 2, we can significantly reduce the computational cost up to 16.9 ∼ 20.8%(-83.1 ∼ -79.2%) compared with the conventional method depending on a given video sequence.

7. Conclusions

In this paper, we have proposed a method to reduce computational cost of the improved super-resolution method by considering the statistics of motion vectors obtained by dense motion estimation (DME) using overlapped block matching (OBM). We have verified that this method can remarkably reduce the entire computational cost up to 29.9 ∼ 49.1% depending on a given video sequence under the condition to completely maintain the original performance of the improved super-resolution method using OBM. Also, we have tried to further reduce computational
cost by relaxing the complete original performance preservation requirement. As a result, we could further reduce computational cost up to 16.9 ∼ 20.8% without serious deterioration of the quality of reconstructed images.

As a future work, we should investigate the performance of our method when we reconstruct images from moving pictures recorded in real circumstance such as monitoring camera.

References


(Received December 29, 2007)