EURASIP Journal on Applied Signal Processing 2004:11, 1708–1720

Multiresolution Motion Estimation for Low-Rate Video Frame Interpolation

Hezerul Abdul Karim
Faculty of Engineering, Multimedia University, 63100 Cyberjaya, Selangor, Malaysia
Email: hezerul@mmu.edu.my

Michel Bister
Division of Engineering, The University of Nottingham, Malaysia Campus, Wisma MISC, 50450 Kuala Lumpur, Malaysia
Email: michel.bister@nottingham.edu.my

Mohammad Umar Siddiqi
Faculty of Engineering, Multimedia University, 63100 Cyberjaya, Selangor, Malaysia
Email: umar@mmu.edu.my

Received 22 October 2003; Revised 18 February 2004; Recommended for Publication by C. C. Jay Kuo

Interpolation of video frames with the purpose of increasing the frame rate requires the estimation of motion in the image so as to interpolate pixels along the path of the objects. In this paper, the specific challenges of low-rate video frame interpolation are illustrated by choosing one well-performing algorithm for high-frame-rate interpolation (Castango 1996) and applying it to low frame rates. The degradation of performance is illustrated by comparing the original algorithm, the algorithm adapted to low frame rate, and simple averaging. To overcome the particular challenges of low-frame-rate interpolation, two algorithms based on multiresolution motion estimation are developed and compared on objective and subjective basis and shown to provide an elegant solution to the specific challenges of low-frame-rate video interpolation.

Keywords and phrases: low-rate video frame interpolation, multiresolution motion estimation.

1. INTRODUCTION

Low frame rates, for example 10 frames per second (fps), are of interest in low-bit-rate video compression. Reducing the frame rate to 5 fps and interpolating it back to 10 fps at the receiver helps to reduce the transmission bit rate. In order to achieve very low data rates for video communication such as in plain old telephone service, it is necessary to skip images at the transmitter, which then have to be reconstructed at the receiver end. The reconstruction of the images can be achieved through frame interpolation. Interpolation of video frames simply means inserting or adding new frames between the video frames. Given the previous and next frames, the task is to insert a new frame between the two. In general, frame interpolation can be performed as illustrated in Figure 1.

Without motion estimation, video frames can be interpolated by averaging the previous and next frames or by repeating the previous frame. The performance of frame interpolation will improve if motion estimation is included in the process. Motion estimation is used to estimate the motion vectors between frames. Pixels are then interpolated along the path of the motion vectors.

In recent years a number of frame interpolation algorithms have been developed [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. Most of them concentrate on high-frame-rate video as shown in Table 1 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13] and part of [16]. In such cases, motion estimation can be achieved by simple block-matching technique. The difference between the algorithms lies mainly in block size, search space, and search technique. Much less work has been done on low frame rates ([14, 15] and part of [16]).

In [14], transmitted motion vectors and segmentation information from an object-based video codec are used to interpolate the skipped frames. In the algorithms presented in this paper, the transmitted motion vectors are not used because for frame interpolation, true motion vector is needed for every pixel [17]. The transmitted motion vectors are not the true motion vectors and they are also produced for every (16×16)- or (8×8)-pixel block as in the case of H.263 video coding standard. Hence, only the previous and next frames are utilized to perform the frame interpolation.

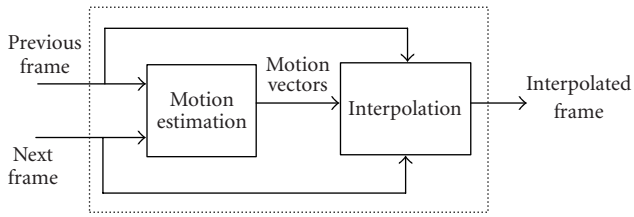

Figure 1: Frame interpolation algorithm.

Table 1: Frame rates for frame interpolation.

Frame interpolation algorithms	Frame-rate conversion (fps)
[1]	50 to 75
[2, 3]	24/30 to 60
[4, 5]	10 to 30
[6, 7, 8, 9]	15 to 30
[10]	10 to 30 and 15 to 30
[11, 12]	50 to 100
[13]	12.5 to 25
[14]	7.5 to 30
[15]	3.75 to 15
[16]	10 to 30, 6 to 30, 12.5 to 50, and 8.3 to 50

A frame interpolation scheme for talking head sequence was proposed in [15]. The foreground and background of previous and next frames are segmented. The foreground area is interpolated using active mesh frame interpolation and the background area is interpolated using averaging. Other areas are interpolated using covered and uncovered technique. Detection and rendition of lip movement is utilized to provide synchronization of lip motion and voice of a person. A good result is produced for simple talking head sequences. Problems may arise if sequences other than talking head are used. The use of multiresolution motion estimation (MRME) for low-rate frame interpolation was investigated in [16, 17]. However, it did not consider the problems that arise in the multiresolution pyramid method due to its rigid structure [18].

In this paper, the basic concept of motion estimation in frame interpolation is described first. A brief review of multiresolution algorithms is presented next, followed by the frame interpolation challenges in low-rate video frames. The algorithms used for frame interpolation are then explained, including the two algorithms based on multiresolution. One of the multiresolution algorithms managed to overcome the rigid pyramid-like structure problem. Results for 5 to 10 fps frame interpolation, discussions, and conclusions are given at the end of the paper.

2. MOTION ESTIMATION

Motion estimation is a process of determining the movement of objects within a sequence of image frames. Block matching is a widely used technique for translation movement estimation. In this method, a block of pixels from frame $n-1$ (previous frame) is matched to a displaced block of pixels in the search area in frame $n+1$ (next frame). The block that gives minimum matching error will be assigned a displacement value called motion vector.

The minimum matching error can be calculated using criteria like mean square error (MSE) and mean absolute difference (MAD) as follows:

$$\text{MSE} = \frac{1}{B_x \times B_y} \sum_{x=-(B_x-1)/2}^{+(B_x-1)/2} \sum_{y=-(B_y-1)/2}^{+(B_y-1)/2} \left[f(x-dx, y-dy, t_{n-1}) - f(x+dx, y+dy, t_{n+1}) \right]^2,$$

$$\text{MAD} = \frac{1}{B_x \times B_y} \sum_{x=-(B_x-1)/2}^{+(B_x-1)/2} \sum_{y=-(B_y-1)/2}^{+(B_y-1)/2} \left| f(x-dx, y-dy, t_{n-1}) - f(x+dx, y+dy, t_{n+1}) \right|, \tag{1}$$

where $f(x-dx, y-dy, t_{n-1})$ is the pixel value at coordinates $(x-dx, y-dy)$ in the $(n-1)$th frame, $f(x+dx, y+dy, t_{n+1})$ is the pixel value at coordinates $(x+dx, y+dy)$ in the $(n+1)$th frame, B_x is the vertical size of the pixels block, and B_y is the horizontal size of the pixels block. For both MSE and MAD, $|dx| \le (S_x - 1)/2$, $|dy| \le (S_y - 1)/2$, where S_x is the vertical size of the search area and S_y is the horizontal size of the search area.

MAD is the most commonly used criterion [19] and has lower complexity compared to MSE [20]. Hence, MAD is used for the block-matching algorithms in this paper.

The motion vector or the displacement (dx, dy) that gives minimum MAD is then obtained as follows:

$$(dx_m, dy_m) = \underset{\substack{|dx| \le S_x \\ |dy| \le S_y}}{\arg\min} \left(\text{MAD}(dx, dy) \right). \tag{2}$$

The selected motion vector (dx_m, dy_m) is then used to interpolate the missing pixel in frame n (frame to be interpolated) as follows:

$$f(x, y, t_n) = \frac{f(x-dx_m, y-dy_m, t_{n-1}) + f(x+dx_m, y+dy_m, t_{n+1})}{2}. \tag{3}$$

3. MULTIRESOLUTION MOTION ESTIMATION

The multiresolution approach in motion estimation techniques is meant to benefit from the divide-and-conquer

capabilities of multiresolution pyramid structures in order to beat the combinatorial explosion problem (see Section 4) of block-matching techniques at a single level of resolution. Global (and large) motion is first estimated at a coarse level of resolution with reduced sampling rate as allowed by the Nyquist criterion. The results of the coarse-level estimates are then propagated to successively higher resolution levels (higher sampling rates) by taking the motion evaluated at the coarse level as an initial estimate for the motion at the next level. This is done iteratively until the full-resolution level is reached. Not only does the MRME approach reduce the computational time in comparison with the single-resolution block-matching methods, but it also achieves better picture quality. Many variations of the MRME are available in the literature and are henceforth discussed according to a general classification into six categories.

(i) *MRME in wavelet video encoder*. Researchers in [21] have initiated an approach of video coding based on MRME at the encoder side. Their method uses the wavelet transform to decompose a video frame into a set of subframes with different resolutions. MRME is utilized to estimate motion at different resolutions of video frame.

(ii) *MRME in subband video coding*. In [22], the frames of the input video are split into seven spatial subbands. Hierarchical motion estimation is used to generate motion vectors for each subband. The initial motion vectors are estimated only in band 1, and are scaled to generate motion vectors for other subbands.

(iii) *Top-down approach of MRME*. The application of top-down MRME for motion and disparity estimation was evaluated in [23]. The motion is estimated at a coarse resolution and is refined at higher resolution level(s). It is better compared to the full search method.

(iv) *Bottom-up approach of MRME*. Several researchers investigated the possibilities of the bottom-up approach. In [24], for example, the motion information extracted at high resolution was preserved, and the information at low resolutions was used only selectively to improve the motion estimates.

(v) *Multigrid structure in MRME*. In [25], the multilevel structure is built on a set of grids with different sizes (multigrid structure). So, it is not a multiresolution approach in the classical sense.

(vi) *Threshold in MRME*. In [26], a thresholding technique is applied in a two-level pyramid to reduce the computation time by preventing blocks whose estimated motion vectors give satisfactory motion compensation from further processing.

From the papers mentioned, it can be concluded that the variants of the fundamental MRME technique differ in terms of the representation of the multiresolution images, the levels of the pyramid, the block size and search space at each level, the motion estimation technique at each level, and general data flow (coarse-to-fine or fine-to-coarse).

4. CHALLENGES IN LOW-RATE VIDEO FRAME INTERPOLATION

To illustrate the challenges of interpolation at low-rate video frames, we took a successful algorithm [1] for high frame rate and applied it to low frame rate with and without adapting the parameters. Then, a new contribution is presented.

When the frame rate is reduced by a factor of N, the search space has to be increased. This means $S_x \times N$ and $S_y \times N$. However, this also increases the risk for false matches between unrelated regions in the search space. Therefore the block size has to be increased by N. This means $B_x \times N$ and $B_y \times N$. With these adjustments, the complexity increases by N^4. This large amount of computation has to be performed in the available time, which has increased by N. The increase of complexity by N^4 because of the speed reduction by N is known as the combinatorial explosion problem.

For example, in 50 to 100 fps interpolation, the algorithm has 20 milliseconds to interpolate between frames. For 5 to 10 fps interpolation, the time available to interpolate between frames is 200 milliseconds. It seems ten times easier than 50 to 100 fps because more time is available. However, using the same video sequence, the movement in 5 fps is ten times larger than the movement in 50 fps. Therefore, the search space and block size have to be increased ten times ($S_x \times 10$, $S_y \times 10$, $B_x \times 10$, $B_y \times 10$). So, 10 000 times more computations are needed in ten times longer time interval.

5. OVERVIEW OF ALGORITHMS USED

Several algorithms to interpolate the low-rate video frames are evaluated. These algorithms are Averaging, Castagno [1], and Adapted Castagno. An approach of MRME is developed and extended (EMRME) to overcome observed artifacts.

5.1. Averaging algorithm

The simplest method to do frame interpolation is by averaging. The previous frame and next frame are averaged at every pixel location in the image according to

$$F_n(x, y) = \frac{F_{n-1}(x, y) + F_{n+1}(x, y)}{2}, \tag{4}$$

where (x, y) is the pixel position, $F_n(x, y)$ is the pixel value at coordinates (x, y) in the frame to be interpolated, $F_{n-1}(x, y)$ is the pixel value at coordinates (x, y) in the previous frame, and $F_{n+1}(x, y)$ is the pixel value at coordinates (x, y) in the next frame.

5.2. Castagno algorithm

The block-matching approach of [1] was designed to interpolate frame rate from 50 to 75 fps. It uses block size of 3×5 and the search space is limited to 5×9 ($(-2, \ldots, +2)$ vertically and $(-4, \ldots, +4)$ horizontally) for the first pixel of every row. For other pixels, the motion vector is found by searching around a 3×3 neighborhood of the previous pixel's motion vector. Hence, a total of nine motion vectors are evaluated

using weighted MAD. The MAD is weighted according to

$$\mathrm{MAD}_{\mathrm{weighted}} = \mathrm{MAD} \times (1 + K \times (dx^2, dy^2)), \quad (5)$$

where K is the elastic constant and the range is suggested to be within 0.05 to 0.02. This formula is discussed in [1]. Basically, it allows the motion vector to move to extreme positions only when it is necessary, and tries to bring it back to a more neutral configuration as often as possible. The motion vector that gives minimum weighted MAD is chosen to be the motion vector for the current pixel. This is repeated for every pixel. When all the pixels' motion vectors have been estimated, a motion vector field is produced, which is postprocessed using 3×3 median filter to remove inconsistent motion vectors. Interpolation is performed based on these motion vectors using the averaging formulas presented in [1], which is slightly different from (3).

5.3. Adapted Castagno algorithm

To adapt the method of [1] to lower frame rates, the following modifications were introduced: (1) instead of evaluating only nine motion vectors, all of the motion vectors in the search range are evaluated for all pixels; (2) the block and the search range are adapted to a low frame rate of 5 fps, which is ten times lower than the frame rate of Castagno. The search space is increased to $(-25, \ldots, +25)$ vertically and $(-45, \ldots, +45)$ horizontally, about ten times the original search space of Castagno. The block size to do block matching is set to 9×15 pixels, three times the original block size of Castagno, which should be ten times according to the explanation before.

However, it was found that increasing the block size further than this did not improve the results and it is reasonable for the size of the search space. This setting is achieved by calculating the fastest motion between previous and next frames of a sequence (e.g., Susie sequence). Clearly, the computational complexity for this method is higher than before as more motion vectors need to be evaluated (4641 motion vectors for every pixel). In this method, weighting is not implemented. Weighting of the MAD makes the algorithm favor vectors that are closer to $(0, 0)$. Hence, it will not choose the large motion vector. Favoring motion vector (0,0) makes the performance of the algorithm almost similar to Averaging algorithm during the frames with fast motion.

5.4. Multiresolution motion estimation (MRME) algorithm

In the MRME algorithm that we propose, the image is filtered using a 7th-order lowpass filter and subsampled to produce successively reduced-resolution versions. Using QCIF images (176×144 pixels), five levels of resolution are considered, the lowest one having 11×9 pixels. For input images with different size, the number of levels in the pyramid may be different. Motion is estimated at the coarsest resolution first to find the global motion. Block size is 9×9 and full search is used. To improve the consistency of the motion field, it is postprocessed with a 3×3 median filter. This motion field is then used as an initial estimate in the next finer resolution. The search space for each pixel is set to 3×3 around the initial estimate.

The block size is maintained at 9×9 at each level. The estimate produced at each level is again submitted to a 3×3 median filter before passing on to the next level. In this way, the motion vectors are refined through the pyramid levels until the highest resolution at the lowest level. The motion vectors at the finest level are then used to interpolate the luminance levels according to the formulas used in [1].

An advantage of the MRME algorithm is that motion detection at high resolution allows for detection of large motion with small search space on low-resolution images, requiring only a small block size. In the high-resolution image, a small search space is also used around the previous estimate, hence also requiring only a small block size.

5.5. Extended multiresolution motion estimation (EMRME) algorithm

5.5.1. EMRME and MRME algorithms formulation

Problems arise in the multiresolution pyramid algorithm due to its rigid structure, as has been documented in [18]. The same problems arise in MRME, and can be alleviated by making the search space extended and adaptive. An example is given in Figure 2, which illustrates an object sliding from left to right. Previous and next frames are visualized in a particular region of interest. To visualize the pyramid better, only two levels have been represented.

We assume that on the higher level (level $k + 1$), the movement is already estimated and median filtered as $dx(k+1, x, y)$, $dy(k+1, x, y)$. At level k, the initial estimate for the motion vector is twice the motion vector and level $k + 1$: $dx_0(k, x, y) = 2 \times dx(k+1, x, y)$ and $dy_0(k, x, y) = 2 \times dy(k+1, x, y)$.

For MRME, the search range is 3×3 around the initial estimate: $dx_0(k, x, y) - 1 \le dx(k, x, y) \le dx_0(k, x, y) + 1$ and $dy_0(k, x, y) - 1 \le dy(k, x, y) \le dy_0(k, x, y) + 1$. For EMRME, the search range is made adaptive based on the previously used 3×3 space and extended to encompass the initial motion estimates of the four neighboring pixels as follows:

$$\begin{aligned} &\min\left(dx_0(k, x, y) - 1, dx_0(k, i, j)\right) \\ &\qquad \le dx(k, x, y) \le \max\left(dx_0(k, x, y) + 1, dx_0(k, i, j)\right), \\ &\min\left(dy_0(k, x, y) - 1, dy_0(k, i, j)\right) \\ &\qquad \le dy(k, x, y) \le \max\left(dy_0(k, x, y) + 1, dy_0(k, i, j)\right), \end{aligned} \quad (6)$$

where $x - 1 \le i \le x + 1$ and $y - 1 \le j \le y + 1$.

5.5.2. EMRME and MRME example

In the example shown in Figure 2, the movement is already estimated and median filtered on the higher level (level $k+1$). The dark pixel is estimated to move by 2 units in horizontal direction, so the movement vector is $(2, 0)$. Other pixels have $(0, 0)$ motion vectors except the two pixels marked with

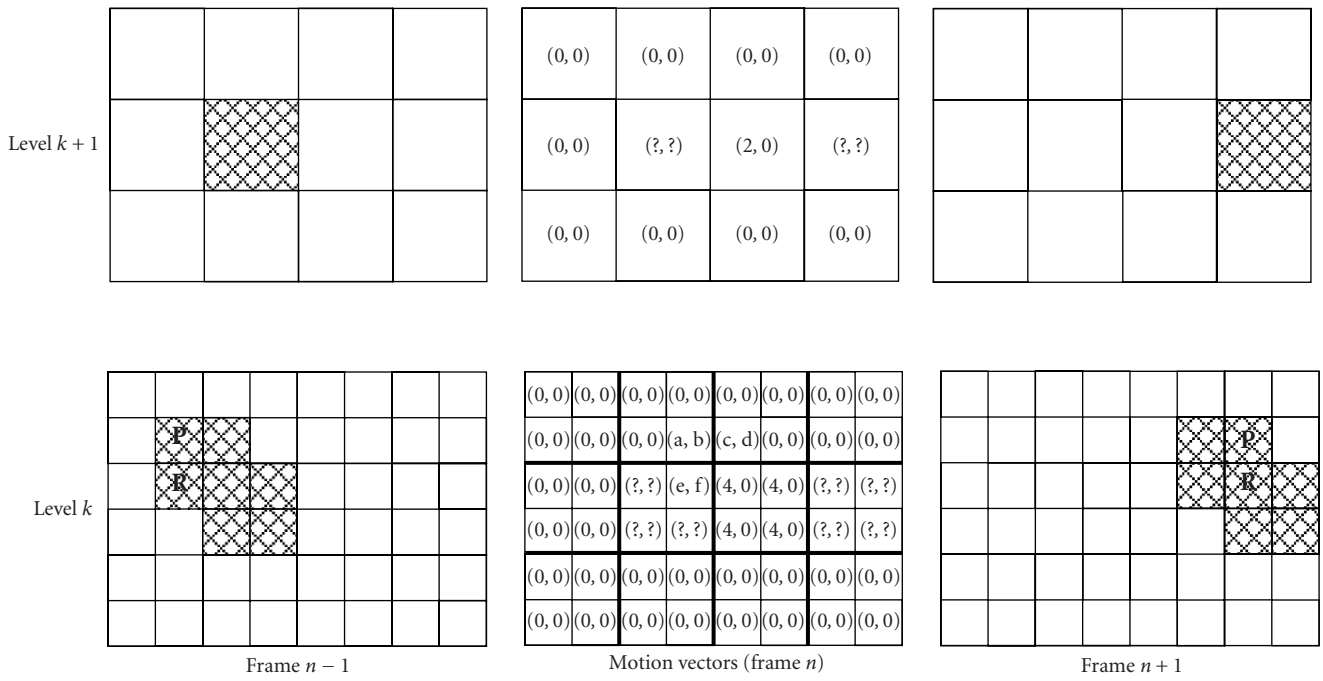

Figure 2: EMRME illustration.

(?,?), which are covered and uncovered pixels. The lower level (level k) has twice the number of pixels of the higher level. One pixel at level $k+1$ corresponds to four pixels at level k. So, the initial motion vectors at level k are equal to the motion vectors of level $k+1$ multiplied by two.

Motion estimation is performed based on the initial estimate. Motion vectors shown in Figure 2 for level k are the resulting motion vectors after the motion estimation on the initial estimate. For four black pixels in the middle, the initial estimate is $(4, 0)$. In case of MRME, the search range is 3×3 around this initial estimate on a motion search space $(-1 \le dx \le 1, -1 \le dy \le 1)$. So, for example, the motion vector candidates for pixel R are $(3, -1)$, $(4, -1)$, $(5, -1)$; $(3, 0)$, $(4, 0)$, $(5, 0)$; $(3, 1)$, $(4, 1)$, $(5, 1)$. Motion vector $(4, 0)$ will be chosen as its match.

5.5.3. Problem in MRME

The problem occurs for pixel P, who has initial motion vector of $(0, 0)$. The correct motion vector for pixel P is $(4, 0)$. The motion vector search space is $(-1 \le dx \le 1, -1 \le dy \le 1)$, hence, the motion vector candidates are $(-1, -1)$, $(-1, 0)$, $(-1, 1)$; $(0, -1)$, $(0, 0)$, $(0, 1)$; $(1, -1)$, $(1, 0)$, $(1, 1)$. It is clear that this search range is insufficient to match the correct pixel in the next frame. The solution to this problem that has been developed in this work is based on an adaptive search space, based on the previously used 3×3 space, but extended to encompass the initial motion estimates of the four neighboring pixels.

The initial estimate of the motion vector (c,d), for example, has been influenced by its three neighbors who make up the pixel on the next higher level. Other examples are motion vectors (a,b) and (e,f). The rigidity of the pyramid structure did not allow for a border between white and black pixels to be located at any other place but at the limits between the next higher pixels. The problem occurs because the movement of pixel P is larger than the 3×3 search space. This measurement was well detected at lower resolution levels. However at this low resolution the border could not be accurately located because it is misaligned with the pyramid grid by one pixel. The original 3×3 search space of pixel P made it impossible to match it with its corresponding pixel in the next frame.

5.5.4. Solution by EMRME

However, now we extend the search space of pixel P to ensure that the initial estimates of the four neighboring pixels are included. In the illustration shown in Figure 2, the search space for pixel P is extended to make it a rectangle that encompasses the initial estimate of the motion vector for neighboring pixels, including pixel R. So, the search space for pixel P is extended from $(-1 \le dx \le +1, -1 \le dy \le +1)$ to $(-1 \le dx \le +5, -1 \le dy \le +2)$. Hence, the correct motion vector for pixel P, $(4, 0)$, is included in this new search space. With this extended search space, it is indeed possible to find a motion vector that matches the pixel P with its match in the next frame. The motion vectors at the finest level are then used to interpolate the luminance levels according to the formulas used in [1].

There could be a hole in the interpolated frame when the motion vectors between the two decoded frames do not

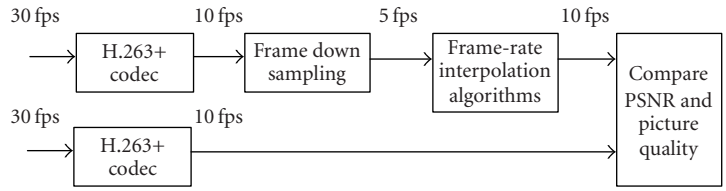

Figure 3: Simulation setup.

pass a pixel in the interpolated frame, that is, when the motion vectors with one or both odd components (e.g., $(3,0)$ and $(3,1)$) have been estimated. Performing a spatial interpolation by a factor of two, vertically and horizontally, of the two input frames is a possible solution. The interpolation formulas in [1] solve this problem by using only original pixels in the motion estimation procedure and the spatially interpolated pixels are only used in the actual interpolation.

As a summary, for both MRME and EMRME algorithms, the search space for each pixel is initially set to 3×3, but for EMRME it is then extended to encompass the initial estimates of its neighbors. In this way, pixels that are located just at a (rigid) boundary of the pyramid structure do not get isolated in their movement from the pixels that are just on the other side of the boundary, and the rigidity of the pyramid structure is overcome.

6. RESULTS AND DISCUSSIONS

The results for the algorithms are compared objectively and subjectively using a simulation setup as shown in Figure 3.

The original image sequences at 30 fps is compressed using H.263+ to 10 fps. The quantization parameter in the H.263+ encoder is fixed to default value of 13 and variable bit rate is used. Downsampling is performed on the 10 fps to get the 5 fps image sequence. The 5 fps is then interpolated to 10 fps using the algorithms discussed. The interpolated 10 fps is finally compared with the originally compressed 10 fps in terms of peak signal to noise ratio (PSNR) and subjective picture quality. The PSNR for comparison is calculated as follows:

$$\text{PSNR} = 10\log_{10}\frac{255^2}{(1/N_xN_y)\sum_{x=0}^{N_x}\sum_{y=0}^{N_y}\left[f_{\text{org}}(x,y)-f_{\text{int}}(x,y)\right]^2}, \tag{7}$$

where $f_{\text{org}}(x,y)$ is the pixel value of the original decoded frame at position (x,y), $f_{\text{int}}(x,y)$ is the pixel value of the interpolated frame at position (x,y), N_x is the vertical size of frame, and N_y is the horizontal size of frame. Simulations are performed on a large number of image sequences that have different sizes. The sizes are 176×144 (QCIF), 352×240 (SIF), and 352×288 (CIF). In Figure 3, frame interpolation is performed after the H.263+ decoder with the aim to reduce jerkiness of the motion in the decoded 5 fps image sequence by interpolating it to 10 fps. Experiments on frame interpolation using uncompressed image sequence have also been done and are reported in [27]. Simulation set up in Figure 3 is reasonable because the main objective is to simulate the decoded 5 fps image sequence and interpolate it to 10 fps.

6.1. Typical results

Figures 4 and 5 show the typical results of interpolation from 5 to 10 fps for QCIF (Claire, Carphone, and Coastguard), SIF (Flower Garden and Football), and CIF (Mobile & Calendar) compressed image sequences. The original image sequences can be downloaded from (http://ise.stanford.edu/video.html) and (http://www.cipr.rpi.edu). Claire is a slow-moving sequence. Carphone, Coastguard, and Mobile & Calendar are fast-moving sequences. Flower Garden and Football are very fast-moving sequences.

Averaging produced overlapped images since it did not take into account the motion between frames. The overlapped image is more obvious in very fast-moving sequences like Flower Garden and Football. Castagno algorithm performs better than Averaging in Claire and Carphone. However, since the block size and search space are too small in Castagno, it fails to interpolate those fast- and very-fast-moving sequences correctly. It is also due to the weighting of MAD in Castagno and hence the algorithm favors $(0,0)$ motion vectors, which produced the same results as Averaging algorithm (see Section 5.3). Although the weighted MAD in Castagno algorithm provides a solution for two overlapping objects between previous and next frames, it does not perform well at low-frame-rate interpolation and in camera tilting such as in the Coastguard sequence. At one stage in the simulation, the weighted MAD was incorporated in our EMRME algorithm, but it did not improve the algorithm since it favors $(0,0)$ motion vectors. However, further simulation studies are needed to confirm this observation.

Adapted Castagno gives sharper images compared to Averaging and Castagno, but there are lots of artifacts due to false matches. The problem of false matches is illustrated for the one-dimensional case in Figure 6. In the image, a gray object is moving from top to bottom. The previous, current (to be interpolated), and next frames are shown. The central pixel in the frame is to be interpolated. It is expected to be interpolated along the dashed movement vector, resulting in a gray pixel value. However, a false match is found with the background, with a movement vector illustrated by the full

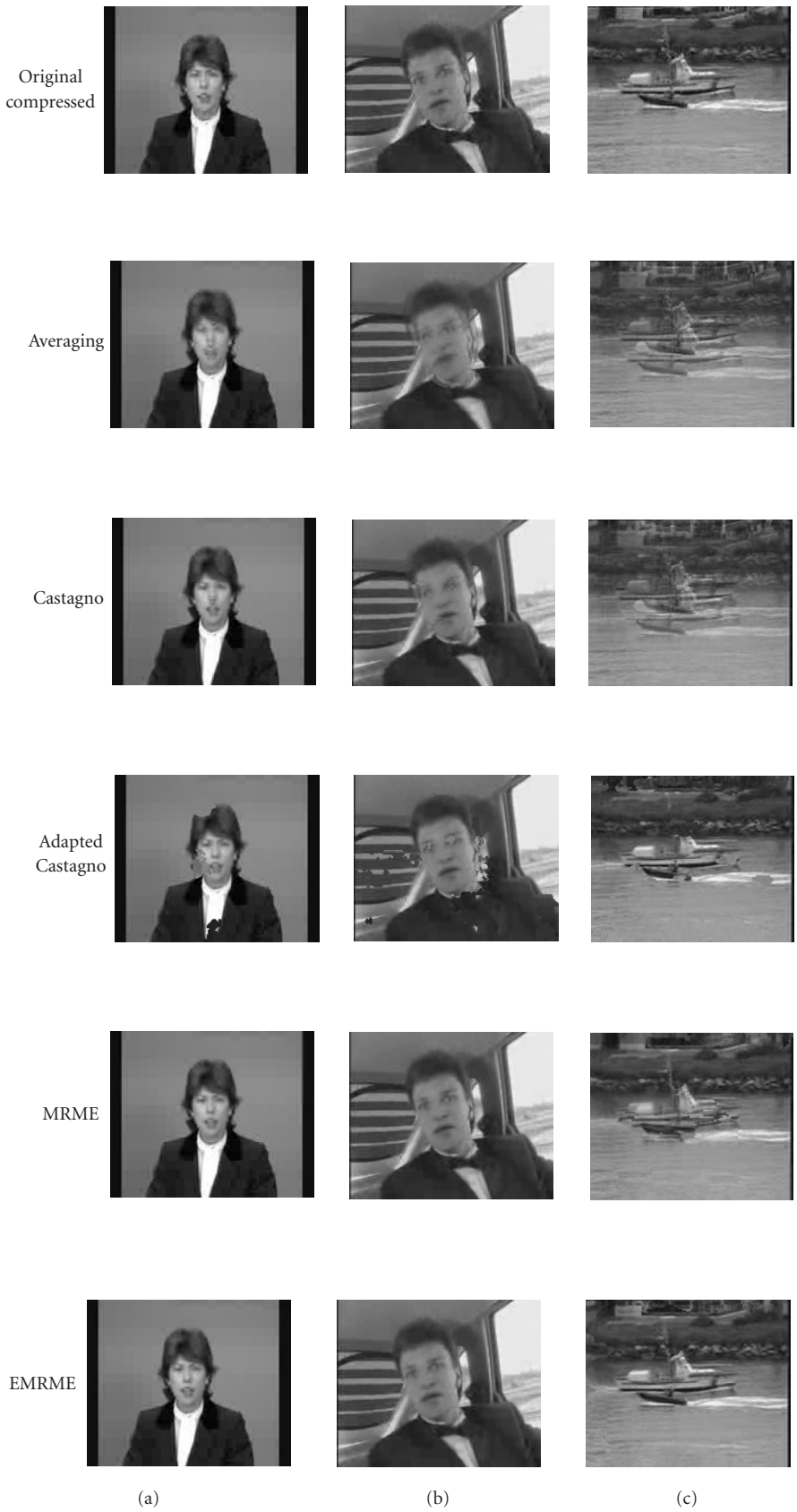

Figure 4: Typical results for interpolation on QCIF image sequences. (a) Claire. (b) Carphone. (c) Coastguard.

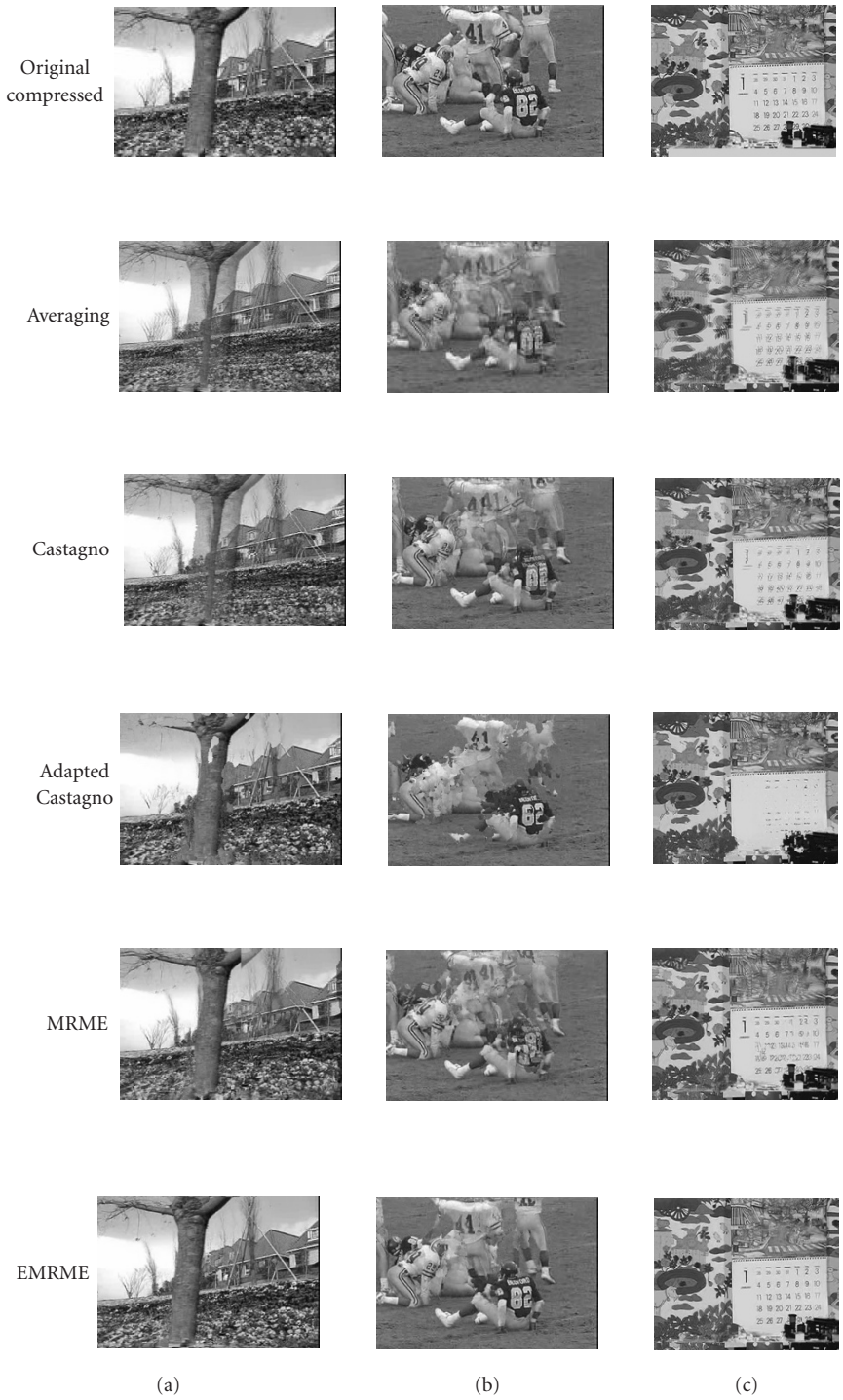

Figure 5: Typical results for interpolation on SIF and CIF image sequences. (a) Flower Garden. (b) Football. (c) Mobile & Calender.

arrow, resulting in a white pixel value. This kind of situation typically occurs with large movements as in the Flower Garden image sequence. It occurs because of the very large block size and search area in Adapted Castagno algorithm, which in turn largely increases the computational time of the algorithm.

Most of the artifacts due to false matches are removed using MRME and EMRME algorithms. Compared to other algorithms, MRME and EMRME perform the best, giving clearer and sharper images. Both remove the averaging and false match artifacts in the interpolated images. At fast- (Coastguard and Mobile & Calendar) and very-fast-moving

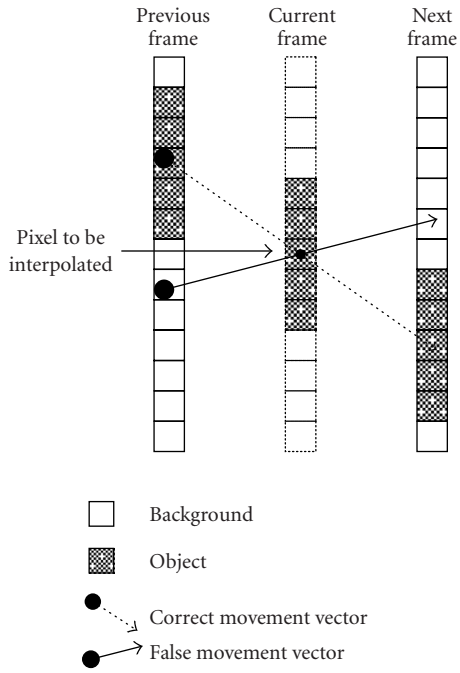

Figure 6: Problem with false matches.

sequences (Flower Garden and Football), EMRME performs better than MRME due to its search space extension and adaptation.

6.2. Quantitative comparison

The objective evaluation (PSNR) of the image sequences interpolated using various interpolation algorithms is shown in Table 2. The best algorithm for each sequence is highlighted in gray in the table and has highest PSNR. For example, for Miss America, EMRME is the best because it has the highest PSNR.

The objective evaluation (PSNR, Table 2) reveals that the EMRME is the best in most cases. Only in a few cases is MRME slightly better. For the Container sequence, Averaging is better—which can be explained by the very slow motion in that particular sequence. For Table Tennis sequence, where the scene has a camera zooming out action, Castagno is slightly better. This probably shows the positive effects of the weighting procedure in Castagno, however it is only for one sequence. Further studies for EMRME are needed to tackle the problem in camera zooming out image sequence. For the Stefan sequence, Adapted Castagno is slightly better. These results are to be expected, since the algorithms were gradually improved, from Castagno to Adapted Castagno to MRME to EMRME, using the PSNR as measuring stick for improvement. The (nice) surprise comes from the fact that the algorithm development was done with one single sequence (Susie), but the results are consistent when using other sequences, and even other formats (SIF and CIF).

6.3. Qualitative comparison

According to [28], at least 15 observers are needed for subjective evaluation of video quality. Therefore, a total of 15 respondents were used for the subjective evaluation of the interpolation. Nineteen image sequences were interpolated using each of the algorithms. The results of the evaluation are shown in Table 3. The algorithms are Averaging, Castagno, Adapted Castagno, MRME, and EMRME.

The interpolated sequences were displayed randomly on computer monitor and respondents were asked to rate the image sequences compared to the original image sequence. The rate is between 0 and 5, with 0 being the worst and 5 about the same as original. The respondents did not know which interpolated sequences corresponded to which algorithm. The original sequence was displayed on the top left corner of the monitor for comparison. The procedure was repeated for each of the image sequences.

The algorithms preferred by the respondents are highlighted in gray in Table 3. For example, for Miss America, EMRME is preferred over the others because the subjective evaluation rate is the highest. This subjective evaluation (Table 3) is more mitigated, although the average is still in favor of the EMRME algorithm. It is to be noted that the Adapted Castagno never comes out as the best algorithm. Averaging and Castagno are preferred in four cases (out of 19), while all the other cases carry MRME or EMRME as preferences. When MRME is preferred, the difference is usually marginal, while when EMRME is preferred, the difference is usually more significant.

The subjective evaluation is less overwhelmingly in favor of EMRME, probably because of the artifacts. PSNR is a kind of "average" error, which does not care about local big errors, as long as the global impression is good, while subjective evaluation is very much affected by local big errors like artifacts. This is illustrated in Figure 7, which shows two corrupted versions of an image of the Susie sequence. In Figure 7a, the error is limited to six pixels which were put to zero, while in Figure 7b the error introduced is a Gaussian error over the whole image with average zero and variance 0.0002. The PSNR is 39.54 and 36.94, respectively. Hence, the PSNR is clearly in favor of the first image, while the subjective evaluation is clearly in favor of the second one. The artifacts introduced in the Adapted Castagno, MRME, and EMRME, are of a nature like the error in Figure 7a, while the blur introduced by the Averaging and Castagno algorithms are more of the nature of the error in Figure 7b.

The global preference, even in the subjective evaluation, is still in favor of the EMRME algorithm, which shows that the improvement introduced, compared to the other algorithms, more than offsets the disturbing artifacts.

6.4. Comparison of computational load of the studied algorithms

The computational load for the EMRME algorithm is much less than for the Adapted Castagno algorithm. Computational load is of the order of $N_x \times N_y \times B_x \times B_y \times S_x \times S_y$,

Table 2: Objective evaluation.

Sequences	Algorithms				
	Averaging	Castagno	Adapted Castagno	MRME	EMRME
(1) Miss America[a]	36.09	38.48	28.50	39.19	39.26
(2) Claire[a]	37.39	38.10	23.00	38.39	38.42
(3) Grandma[a]	40.02	40.75	37.51	41.10	41.10
(4) Akiyo[a]	38.82	39.32	37.63	40.04	40.01
(5) Container[a]	40.18	39.42	33.05	39.31	39.40
(6) Susie[a]	27.99	28.47	27.18	29.35	29.89
(7) Carphone[a]	29.67	30.08	22.74	30.45	30.51
(8) Foreman[a]	25.01	26.09	22.38	26.43	27.27
(9) Salesman[a]	35.00	35.17	30.38	35.53	35.50
(10) Trevor[a]	27.11	27.29	23.27	27.00	27.42
(11) Coastguard[a]	24.51	25.01	23.13	24.60	26.37
(12) News[a]	29.26	29.97	22.88	30.55	30.62
(13) Mother & Daughter[a]	32.43	32.56	29.80	32.61	32.62
(14) Hall-objects[a]	32.49	32.93	27.18	32.55	34.00
(15) Silent[a]	30.31	30.37	27.83	30.52	30.52
(16) Flower Garden[b]	13.72	13.91	16.75	15.76	18.10
(17) Table Tennis[b]	21.56	21.66	21.22	21.33	21.60
(18) Football[b]	17.52	17.63	17.23	17.65	18.02
(19) Stefan[c]	18.21	18.29	19.29	17.71	19.08
Average	16.87	17.01	18.28	17.69	18.99

[a]QCIF (176×144)
[b]SIF (352×240)
[c]CIF (352×288)

where (N_x, N_y) is the image dimension, (B_x, B_y) is the block size, and (S_x, S_y) is the search space. In the case of MRME, the image dimension to be taken into account is the sum of the image dimension at each level, namely, $(144 \times 176) + (72 \times 88) + (36 \times 44) + (18 \times 22) + (9 \times 11)$, with $B_x = B_y = 9$ and $S_x = S_y = 3$. The resulting estimate of calculations to interpolate *one* image is 58 164 480 operations for Castagno, 15 878 903 040 for Adapted Castagno (273 times more than Castagno!), and 24 610 311 for MRME (42% of Castagno). For EMRME, the search space is not fixed and is possibly larger than for MRME, so while the number of operations will be larger than for MRME, no upper limit can be given.

Computational load of Castagno, Adapted Castagno, and EMRME is shown in Table 4. The ratios are 1000 : 1 and 3 : 1 for Adapted Castagno and EMRME compared to Castagno, respectively. Although EMRME is slower possibly due to the fact that the lower-resolution images have to be calculated before starting the motion estimation, it is to be remembered that Castagno was designed for 50 to 75 fps (hence 40 millisecond time to do the calculation). EMRME was designed for 5 to 10 fps (hence 200 millisecond time to do the calculation), hence EMRME is still doing better than Castagno. In this computational load comparison, Castagno algorithm is chosen as the performance metric because the frame interpolation using EMRME is developed based on Castagno algorithm that works well at high-frame-rate interpolation, but not at low-frame-rate interpolation. Performance difference with other real-time frame interpolation schemes, such as [5], can be investigated provided that the EMRME algorithm is optimized for real-time application.

7. CONCLUSIONS

Interpolation at low frame rate is a great challenge. Most existing algorithms interpolate at high frame rate (e.g., Castagno). The algorithms have to be adapted to assess fast motion (resulting in large frame-to-frame displacement) occurring at low frame rate. Classical block matching introduces combinatorial problems. Small block size and small search area cannot detect fast motion (e.g., Castagno). On the other hand, large block size and large search area produce mismatches, which lead to artifacts and speed reduction (Adapted Castagno).

The MRME algorithm is proposed and implemented. It estimates the movement first at lower resolution (smaller search space), and then successively increases the resolution

Table 3: Subjective evaluation.

Sequences	Algorithms				
	Averaging	Castagno	Applied Castagno	MRME	EMRME
(1) Miss America[a]	2.67	2.60	0.73	3.20	3.27
(2) Claire[a]	1.87	1.87	0.73	2.07	2.00
(3) Grandma[a]	2.53	2.07	0.87	2.27	2.40
(4) Akiyo[a]	1.93	2.00	1.40	2.27	2.20
(5) Container[a]	2.93	2.93	0.87	2.47	2.67
(6) Susie[a]	1.80	2.00	1.13	2.67	2.33
(7) Carphone[a]	1.80	1.73	0.67	2.20	2.27
(8) Foreman[a]	1.60	1.60	0.73	1.67	2.07
(9) Salesman[a]	2.07	2.00	0.60	1.80	1.80
(10) Trevor[a]	1.60	1.53	0.40	1.93	1.93
(11) Coastguard[a]	2.67	2.60	0.87	1.87	2.53
(12) News[a]	2.20	2.80	1.00	2.80	2.40
(13) Mother & Daughter[a]	0.93	1.07	0.87	0.60	1.07
(14) Hall-objects[a]	2.40	2.33	0.47	2.27	2.67
(15) Silent[a]	1.80	2.00	1.13	2.67	2.33
(16) Flower Garden[b]	2.60	2.87	1.67	2.33	3.73
(17) Table Tennis[b]	1.53	1.93	1.27	1.60	1.47
(18) Football[b]	2.60	2.20	1.07	2.73	3.13
(19) Stefan[c]	1.07	1.47	0.73	1.67	2.33
Average	1.95	2.12	1.18	2.08	2.67

[a]QCIF(176×144)
[b]SIF(352×240)
[c]CIF(352×288)

(a) (b)

Figure 7: Susie image with (a) strong local error and (b) small global error.

and performs search only in a subset of the search space around the search solution from the previous hierarchical level. It manages to reduce artifacts and long computation time introduced by the Adapted Castagno algorithm.

An improved version of MRME algorithm, called the EMRME algorithm, is also proposed. The EMRME algorithm reduces artifacts further because of its extended and adaptive search space, which is obtained by stretching the search space to include the search space of the current, previous, next, top, and bottom pixels. The EMRME algorithm is superior to Averaging, Castagno, Adapted Castagno, and MRME algorithms, both subjectively and objectively.

Table 4: Computational load comparison for Pentium Pro II 350 MHz, 256 MB RAM.

	Castagno	Adapted Castagno	EMRME
To interpolate 1 image of 176 × 144	2.4 s	40 min	7.2 s
To interpolate 25 images of 176 × 144	1 min	12 h	3 min

REFERENCES

[1] R. Castagno, P. Haavisto, and G. Ramponi, "A method for motion adaptive frame rate up-conversion," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 6, no. 5, pp. 436–446, 1996.

[2] K. A. Bugwadia, E. D. Petajan, and N. N. Puri, "Progressive-scan rate up-conversion of 24/30 Hz source materials for HDTV," *IEEE Transactions of Consumer Electronics*, vol. 42, no. 3, pp. 312–321, 1996.

[3] R. L. Lagendijk and M. I. Sezan, "Motion compensated frame rate conversion of motion pictures," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP '92)*, vol. 3, pp. 453–456, San Francisco, Calif, USA, March 1992.

[4] D.-W. Kim, J.-T. Kim, and I.-H. Ra, "A new video interpolation technique based on motion-adaptive subsampling," *IEEE Transactions of Consumer Electronics*, vol. 45, no. 3, pp. 782–787, 1999.

[5] T.-Y. Kuo, J. Kim, and C.-C. J. Kuo, "Motion-compensated frame interpolation scheme for H.263 codec," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS '99)*, vol. 4, pp. 491–494, Orlando, Fla, USA, May-June 1999.

[6] R. Krishnamurthy, J. W. Woods, and P. Moulin, "Frame interpolation and bidirectional prediction of video using compactly encoded optical-flow fields and label fields," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 9, no. 5, pp. 713–726, 1999.

[7] Y.-K. Chen, A. Vetro, H. Sun, and S. Y. Kung, "Frame-rate up-conversion using transmitted true motion vectors," in *Proc. 2nd IEEE International Workshop on Multimedia Signal Processing*, pp. 622–627, Redondo Beach, Calif, USA, December 1998.

[8] B.-T. Choi, S.-H. Lee, Y.-J. Park, and S.-J. Ko, "Frame rate up-conversion using the wavelet transform," in *Proc. IEEE International Conference on Consumer Electronics-Digest of Technical Papers (ICCE '00)*, pp. 172–173, Los Angeles, Calif, USA, June 2000.

[9] B.-T. Choi, S.-H. Lee, and S.-J. Ko, "New frame rate up-conversion using bi-directional motion estimation," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 3, pp. 603–609, 2000.

[10] W. R. Sung, E. K. Kang, and J. S. Choi, "Adaptive motion estimation technique for motion compensated interframe interpolation," *IEEE Transactions on Consumer Electronics*, vol. 45, no. 3, pp. 753–761, 1999.

[11] O. A. Ojo and G. de Haan, "Robust motion-compensated video upconversion," *IEEE Transactions on Consumer Electronics*, vol. 43, no. 4, pp. 1045–1056, 1997.

[12] G. de Haan, J. Kettenis, A. Loning, and B. De Loore, "IC for motion-compensated 100 Hz TV with natural-motion movie-mode," *IEEE Transactions on Consumer Electronics*, vol. 42, no. 2, pp. 165–174, 1996.

[13] N. Grammalidis, D. Tzovarns, and M. G. Strintzis, "Temporal frame interpolation for stereoscopic sequences using object-based motion estimation and occlusion detection," in *Proc. IEEE International Conference on Image Processing (ICIP '95)*, vol. 2, pp. 2382–2385, Washington, DC, USA, October 1995.

[14] S.-C. Han and J. W. Woods, "Frame-rate up-conversion using transmitted motion and segmentation fields for very low bit-rate video coding," in *Proc. IEEE International Conference on Image Processing (ICIP '97)*, vol. 1, pp. 747–750, Santa Barbara, Calif, USA, October 1997.

[15] T. Chen, Y. Wang, H. P. Graf, and C. Swain, "A new frame interpolation scheme for talking head sequences," in *Proc. IEEE International Conference on Image Processing (ICIP '95)*, vol. 2, pp. 591–594, Washington, DC, USA, October 1995.

[16] R. Thoma and M. Bierling, "Motion compensating interpolation considering covered and uncovered background," *Signal Processing: Image Communications*, vol. 1, pp. 191–212, 1989.

[17] M. Bierling and R. Thoma, "Motion compensating field interpolation using a hierarchically structured displacement estimator," *Signal Processing*, vol. 11, no. 4, pp. 387–404, 1986.

[18] M. Bister, J. Cornelis, and A. Rosenfeld, "A critical view of pyramid segmentation algorithms," *Pattern Recognition Letters*, vol. 11, no. 9, pp. 605–617, 1990.

[19] G. Van der Auwera, G. Lafruit, and J. Cornelis, "Arithmetic complexity of motion estimation algorithms," in *Proc. IEEE Benelux Signal Processing Chapter, Signal Processing Symposium (SPS '98)*, pp. 203–206, Leuven, Belgium, March 1998.

[20] R. J. Clarke, *Digital Compression of Still Images and Video*, Academic Press, London, UK, 1995.

[21] S. Zafar, Y.-Q. Zhang, and B. Jabbari, "Multiscale video representation using multiresolution motion compensation and wavelet decomposition," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 1, pp. 24–35, 1993.

[22] J. Lee and B. W. Dickinson, "Subband video coding with scene-adaptive hierarchical motion estimation," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 9, no. 3, pp. 459–466, 1999.

[23] D. Tzovaras, M. G. Strintzis, and H. Sahinoglou, "Evaluation of multiresolution block matching techniques for motion and disparity estimation," *Signal Processing: Image Communications*, vol. 6, no. 1, pp. 59–67, 1994.

[24] I. Rhee, G. R. Martin, and R. A. Packwood, "A multiresolution block matching technique for image motion estimation," in *IEE Colloquium on Multiresolution Modelling and Analysis in Image Processing and Computer Vision*, pp. 5/1–5/6, London, UK, April 1995.

[25] F. Dufaux and F. Moscheni, "Motion estimation techniques for digital TV: a review and a new contribution," *Proceedings of the IEEE*, vol. 83, no. 6, pp. 858–876, 1995.

[26] Y. Q. Shi and X. Xia, "A thresholding multiresolution block matching algorithm," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 7, no. 2, pp. 437–440, 1997.

[27] H. A. Karim, "A study of motion estimation algorithms for interpolation of low rate video frames," Master's of Engineering Science thesis, Multimedia University, Selangor, Malaysia, 2002.

[28] "Subjective video quality assessment methods for multimedia applications," ITU-T Recommendation P.910, 1999.

Hezerul Abdul Karim received B.Eng. degree in Electronics with Communications from University of Wales Swansea, UK, in 1998. He was an intern at Bell Labs, Lucent Technologies, USA, from October 1999 to March 2000. He received his M.Eng. degree from Multimedia University, Malaysia in 2003. He worked as a Tutor and then as a Lecturer at the university from 1998 to 2003. He is currently on study leave for his Ph.D. degree at Center for Communication Systems Research (CCSR), School of Electronics and Physical Sciences, University of Surrey, UK. His research interests include telemetry, image/video processing, motion estimation, and 3D video compression and transmission.

Michel Bister is an Associate Professor at The University of Nottingham, Malaysia Campus. He worked on computer analysis of cardiac MR images in his Ph.D. degree, then supervised a group on artificial intelligence in diagnostic imaging at the Brussels University (VUB). After a passage in a small company, he joined the Nuclear Medicine Department of a regional hospital (St. Elisabeth Hospital, Zottegem, Belgium), spent a few years developing and improving systems for computer-aided diagnostics (CAD) in SPECT images of heart, brain, and kidney, where he coauthored a patent on a system for analyzing heart function using 4D gated SPECT. Dr. Bister's next step was three years at a private university (Multimedia University) in Malaysia, building up an image processing research group of a dozen persons. He is currently the Chairman of the Research Committee of the Engineering Division, The University of Nottingham Malaysia Campus, where he also chairs the Digital Signal Processing Group.

Mohammad Umar Siddiqi received B.S. Eng. and M.S. Eng. degrees from Aligarh Muslim University (AMU Aligarh) in 1966 and 1971, respectively, and Ph.D. degree from Indian Institute of Technology, Kanpur (IIT Kanpur) in 1976, all in electrical engineering. He has been in the teaching profession throughout, first at AMU Aligarh, then at IIT Kanpur. Currently, he is a Professor in the Faculty of Engineering at Multimedia University, Malaysia. His research interests are in coding and cryptography. He has published more than 75 papers in international journals and conferences.