ISSN 0361-7688, Programming and Computer Software, 2009, Vol. 35, No. 6, pp. 351–364. © Pleiades Publishing, Ltd., 2009.
Original Russian Text © D.S. Vatolin, S.V. Grishin, 2009, published in Programmirovanie, 2009, Vol. 35, No. 6.

Double Up-Conversion of Video Frame Rate Based on Bidirectional Motion Compensation

D. S. Vatolin and S. V. Grishin

Graphics and Multimedia Laboratory, Department of Computational Mathematics and Cybernetics, Moscow State University, Moscow, 119992 Russia

E-mail: dmitriy@graphics.cs.msu.su, grishin@graphics.cs.msu.edu

Received October 20, 2008

Abstract—An algorithm for video sequence frame rate double up-conversion is described. The algorithm is based on bidirectional motion compensation. The algorithm has no smoothing in the time domain implying the absence of the image sharpness oscillations in the transformed video sequence. A special post-processing step with adaptively controlled degree of smoothing makes it possible to considerably decrease the "blocking" artifact while retaining the maximum number of image details. The absence of complicated mathematical computations allows realtime hardware implementation of the algorithm and real–time video processing.

DOI: 10.1134/S036176880906005X

1. INTRODUCTION

Frame rate conversion (FRC) is widely used in digital video processing. Firstly, FRC is required for the transformation of video standards. For example, the frame rate of 25 fps is typical for movies, while the most up–to–date TV sets can reproduce 100 fps.

Another important domain of FRC application is video data decoding. Many video codecs with a high degree of data compression decrease the data resolution both in spatial and temporal domains (in this paper, the frame rate will be also referred to as time resolution). In other words, instead of coding the initial frame rate of 30 fps, it can be reduced to 20 fps. This means that every third frame of the initial video sequence is not coded. Meanwhile, the initial frame rate is desirable when playing the decoded video file. In order to do this, an algorithm for data recovery is required. FRC is used as such an algorithm. FRC can be also used to recover damaged frames.

Another possible domain of FRC application is the enhancement of video visual quality. For example, the frame rate in videos shot using a cell phone or a webcam is usually 15 fps. With such a frame rate, the time discreteness (discontinuous motion) of the video is visually noticeable if the frame contains moving objects. Visual quality of such videos can be considerably improved using up-conversion of the frame rate.

FRC can be regarded as time interpolation of the video. From this point of view, it is analogous to image resampling algorithms (IRA). While IRA are intended to enhance the spatial resolution, FRC deals with the time resolution. Similarly to interpolated pixels in IRA, frames produced by FRC are called interpolated frames (IF). One can say that FRC extends IRA on the temporal axis. Similarly to IRA, the principle of the image edge-directed interpolation (EDI) is extremely important in FRC, though in video the edges have three dimensions (two spatial dimensions plus one temporal dimension). This partially explains the choice of the methods for the edge detection (ED) and

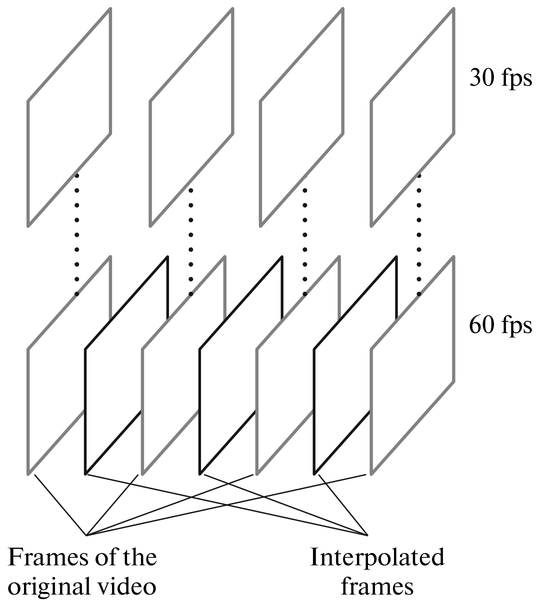


Fig. 1. Scheme for adding interpolated frames.

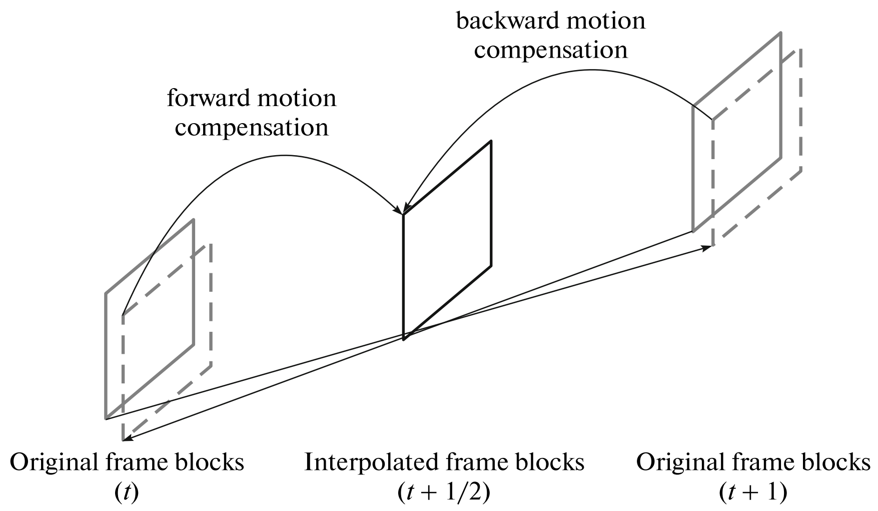

backward motion compensation
forward motion compensation
Original frame blocks (t)
Interpolated frame blocks (t + 1/2)
Original frame blocks (t + 1)

Fig. 2. Bidirectional motion compensation.

EDI in FRC, which are different from the methods used in IRA.

The methods for ED and EDI in IRA (for example, differential methods) use the fact that the distance between two adjacent points of one edge does not exceed one pixel and a half. In the video, this distance can be as large as tens of pixels if the points are situated on the adjacent frames, i.e. when the temporal dimension is considered. This makes it impossible to apply the IRA methods for ED and EDI directly in FRC. We say *directly* because there exists a trick that makes it possible, in most cases, to reduce the distance between two time adjacent points of the same edge. This is motion compensation (MC), which consists in constructing compensated frames (CF). Before MC, motion information (MI) is calculated, which is called motion estimate (ME). Each CF is reconstructed from the frame of the initial video sequence on the basis of MI calculated from ME for the following initial frame.

Let consider the process of motion compensation in detail. Assume that, for every point of some frame from the initial video sequence, ME makes it possible to calculate the difference between the coordinates of this point on the current frame and on the preceding frame. This difference is referred to as the motion vector. Then, the currently compensated frame is the frame constructed from the initial one by shifting each point by its motion vector. Note that each compensated frame is associated with a point of time for which it is compensated. Thus, in the pair (current frame, CF), the distances between the time adjacent points of the same edge are considerably reduced compared with the pair (current frame, preceding frame). Note that, in practice, motion vectors in some areas of the frame cannot be found with sufficient accuracy. This makes it necessary to estimate the accuracy of the MI and to change the application application strategy of the ED and EDI algorithms accordingly. This is one of the differences between FRC and IRA.

The simplest method for recovering the missing frames is to repeat the frame that precedes the missing one (frame repetition, FR). The main disadvantage of this method is the motion discontinuity is still noticeable in the transformed video sequence because there is no edge-directed interpolation. Since the enhancement of the time resolution is the main purpose of the FRC algorithm, the scope of the FR method, which does not solve this problem, is quite small. Another method is linear interpolation between the encoded frames. But this method leads to halos on the edges of the moving objects. This phenomenon is usually called ghost artifact. Besides, just like in the previous method, there is no edge-directed interpolation.

A considerable improvement in the quality is achieved by the algorithms using the edge-directed interpolation (EDI). As noted above, in order to improve the efficiency of EDI, it is necessary to estimate motion parameters between the video frames. ME can be performed in several ways. The majority of FRC algorithms use the ME methods based on block matching search (block matching algorithms, BMA) [1–4], which are computationally efficient and hardware implementable. A disadvantage of these methods is the potential possibility of the artifact called “blockiness” on the CF. Presently, there exist a number of FRC algorithms that use information about the motion. For example, the paper [1] offers a method for one and a half time up-conversion of the frame rate. The method described in that paper arranges the interpolated frames in time so that all the initial frames in the converted sequence can be used and only divisions by the powers of two are needed, which considerably facilitates the hardware implementation. However, the time interval between the initial frames is disrupted in this method, which leads to nonuniform motion ("jerkiness"). The paper [2] describes a method for recovering the missing frames at the side of the decoder H.263. The interpolated frames are calculated using a hybrid scheme that adaptively selects the interpolation method—the frame repetition or compensated frame interpolation. A disadvantage of this method is that its scope is restricted by one type of decoder. The method for the interpolation of the compensated frames described in [4] is developed for the decoder ITU–T H.263 that deals with the sequences including M–frames (motion frames). This method does not require the motion between the base frames to be uniform and rectilinear—the information about the "nonlinear vectors" of the motion is stored by the video coder in the M–frames and is used for decoding. That is why the application of this method is also restricted to the decoder H.263. The procedure of the global motion compensation (GMC) proposed in [5] as a method for constructing the compensated frames makes it possible to correctly process the intensive motions of a special type (rotation, zoom–in and zoon–out of the camera). However, the GMC procedure is insensitive to the motion of local objects, which leads to the discontinuous motion of some object in the frame. In [6], an Optical Flow Estimation Algorithm is used for the computation of the motion vectors. The scheme based on label fields is proposed for processing occlusions. This scheme gives good results; however, the calculation of the label fields requires information about the frame being interpolated, which restricts the use of this method to the decoder side.

The algorithm proposed in this paper is developed for double up-conversion of the frame rate. In order to estimate the motion, we use the method based on the calculation of the correct motion vectors. The main difference between this method and other ME methods is its orientation the search of the motion vectors that both minimize the frame difference function (that is typical for the conventional ME methods) and have a correct direction (coinciding with the real motion direction in this area). The compensated frames are calculated with the accuracy of one fourth of a pixel (obtained by using the frames zoomed four times in the process of the motion compensation), which provides high accuracy in expressing weak motion and slowly changing areas. Motion compensation is performed after the ME stage on the basis of the information obtained from ME. The details of the compensated frames calculation are described in Section 2.

The main feature of the proposed method is the fact that the converted sequence has a time–constant sharpness level; i.e., the interpolated frames and the initial frames have the same sharpness. This is not typical for the FRC algorithms because time–average the pixels' color parameters are usually over time. An additional advantage of the proposed method is the possibility to process frames without a priori information about the interpolated frames, which augments the its scope.

2. ALGORITHM DESCRIPTION

In order to shorten the description of the algorithm and to make it more clear, only the brightness component of the video sequence is considered. The color components are processed simultaneously with the brightness operations.

The doubling of the frame rate is achieved by addition of the interpolated frames (IF) between every two adjacent frames of the initial sequence (see Fig. 1).

The operation of the algorithm is an iterative process of calculating the interpolated frames and adding them to the initial video sequence. The converted video sequence $\hat{I}(p, n)$ can be written as

$$\hat{I}(p,n) = \begin{cases} I_0(p,k), & n = 2k,\ k \in \{0,1,2,\ldots\}, \\ \tilde{I}(p,m), & n = 2m+1,\ m \in \{0,1,2,\ldots\}, \end{cases} \quad (1)$$

$$\tilde{I}(p,m) = \begin{cases} C_F(p,m), & \bar{E}_F(p) \le \bar{E}_B(p), \\ C_B(p,m), & \bar{E}_B(p) < \bar{E}_F(p), \end{cases} \quad (2)$$

$$\begin{aligned} C_F(p,m) &= I_0(p + \hat{v}_F(p,k), k), \quad m = 2k+1, \\ C_B(p,m) &= I_0(p + \hat{v}_B(p,k), k+1), \quad m = 2k+1, \end{aligned} \quad (3)$$

$$\hat{v}_F(p,k) = \begin{cases} \dfrac{v_F(p,k)}{2}, & e_F(p) < TH, \\ 0, & e_F(p) \ge TH, \end{cases} \quad (4)$$

$$\hat{v}_B(p,k) = \begin{cases} \dfrac{v_B(p,k)}{2}, & e_B(p) < TH, \\ 0, & e_B(p) \ge TH, \end{cases} \quad (5)$$

where

$p(x, y)$ are the pixel coordinates in the frame;

$\hat{I}(p, n)$ is the brightness of the pixel p in the frame n in the converted sequence;

$I_0(p, k)$ is the brightness of the pixel p in the initial frame k;

$\tilde{I}(p, m)$ is the brightness of the pixel p in the interpolated frame m;

C_F, C_B are the forward and backward compensated frames, respectively;

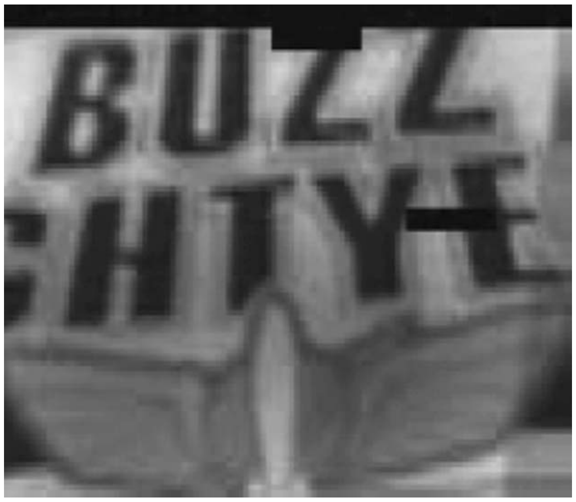


Fig. 3. Frame fragment from the toy sequence after the step 2.

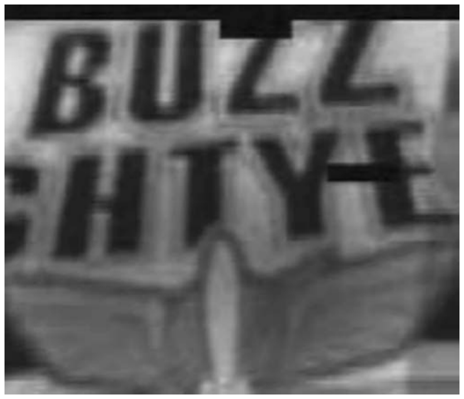


Fig. 4. Before the step 3.

v_F, v_B are the motion vectors calculated for the blocks of the interpolated frame based on the information obtained at the forward and backward stages of ME; the calculation method for these vectors is described in detail in section 2.1;

e_F, e_B are the compensation errors of the vectors v_F and v_B calculated for the blocks of the compensated frame; the calculation method for these functions is described in detail in section 2.1;

TH is a fixed threshold;

$\bar{E}_F$, $\bar{E}_B$ are the average values of the errors in the neighborhood of radius 1 around the pixel p in the array of pixel errors (see Fig. 10) in the forward or backward compensated frame, respectively.

It can be seen from formulas (2) that each pixel of the interpolated frame is a pixel of one of the compensated frames (CF). Each CF consists of the pixels of the initial video frames shifted by the corresponding motion vectors (3). The motion vectors for the 4×4 blocks of the compensated frames are calculated on the basis of the information obtained at the ME stage. The calculation algorithm for these vectors is described in detail in section 2.1. Thus, two CF are used to construct one IF. We will call these frames forward or backward compensated frames depending on the direction of the ME stage at which the motion information used for the compensation is obtained (see Fig. 2). The compensated frames are calculated for the point of time of the corresponding interpolated frame, which is equally distant in time from the adjacent frames of the initial video. This explains the division by 2 of the vectors in formula (4) (see Fig. 2).

Thus, each CF is constructed on the basis of two frames of the initial sequence are before and after the required IF in time (see Fig. 8). We will call them key frames (KF). The first stage of CF construction is the motion estimation (ME). Let us consider how the motion estimation is performed. For each block in one of the KF (referred to as main KF, MKF), we search for the most similar block in the second KF (referred to as reference KF, RKF). Similarity of the blocks is measured, for example, by the sum of the absolute differences (SAD) of the brightness values of the points of these blocks. The value of the similarity measure for a given vector and a given block is referred to as the compensation error of this block. Forward direction of the compensation corresponds to the case when MKF is later in time than the RKF. Conversely, the MKF in backward compensation is earlier in time than the RKF.

Taking into account the above definitions, the basic steps of the CF calculation can be written as follows (see Fig. 9).

1. **Motion estimation.** Calculation of the motion vectors in specific direction (forward or backward).

2. **Filling the object area.** Filling the blocks of the compensated frame that have an admissible (below the threshold) value of the compensation error (see Fig. 3). Calculation of the array of pixel errors of the compensated frame.

3. **Processing the motion on the frame borders.** Search and processing of the empty areas near the borders of the compensated frame (see Figs. 4 and 5) appeared as a result of the motion nonparallel to the border (i.e., the motion with a nonzero normal component).

4. **Correction of the filled area.** Search and filling of the empty blocks in the compensated frame that are not parts of the occlusion areas (see step 5). Results of this step can be seen in Figs. 6 and 7.

5. **Filling the occlusion areas.** Occlusion areas are the parts of the frame that are seen only in one of the two key frames. Such areas can be located, for example, near the edges of the moving objects. It is assumed that the only areas that remain empty before performing this step are the occlusion areas that are filled using image data from the preceding or the next key frame.

6. **Deblocking.** Since the compensated frames are calculated block by block, they can have a defect called *blockiness*. In order to remove it, a deblocking procedure is performed at the last step of the compensated frame construction. Blockiness is suppressed by blurring the image in the direction perpendicular to the borders of the 4 × 4 blocks. The intensity of blurring is determined adaptively and depends on the characteristics of the image neighborhood being processed. Thus, the image is blurred stronger in the places with a high blockiness level, and it is not changed in the areas containing object edges or contrast textures.

Each step of the construction of the compensated frame, except for the motion estimation, is described in detail below. The ME algorithm is not considered in the present paper because it is not connected directly to the problem of the frame rate conversion. A description of various ME algorithms can be found in [7].

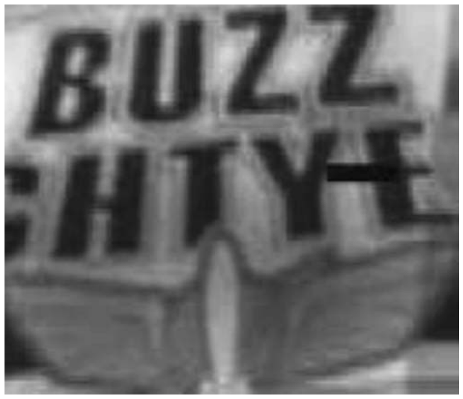


Fig. 5. After the step 3.

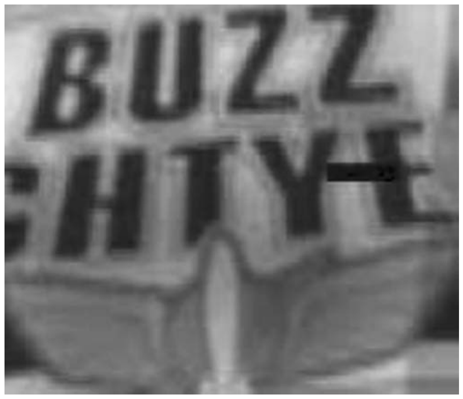


Fig. 6. Before the step 4.

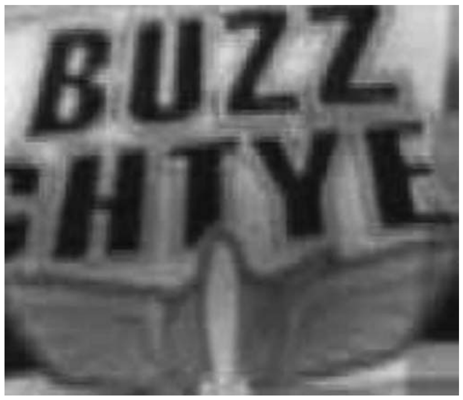


Fig. 7. After the step 4.

2.1. Filling the Object Area

At this step, blocks of size 4 × 4 are processed. We will call them M–blocks (main blocks). Only the M–blocks are processed for which the motion vector was found with an admissible (below the threshold) compensation error. The following steps are performed for each of such blocks:

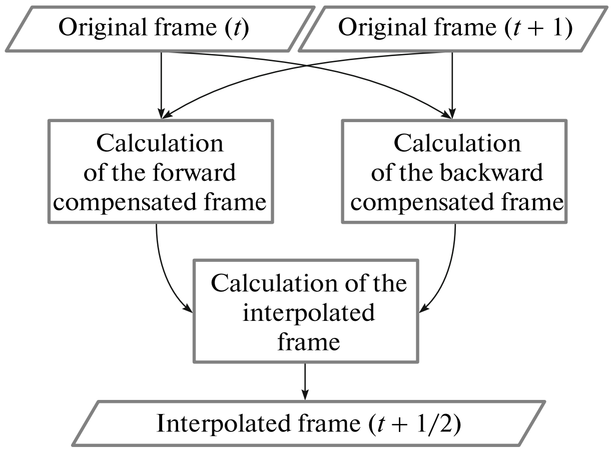

Original frame (t)
Original frame (t + 1)
Calculation of the forward compensated frame
Calculation of the backward compensated frame
Calculation of the interpolated frame
Interpolated frame (t + 1/2)

Fig. 8. Interpolated frame construction scheme.

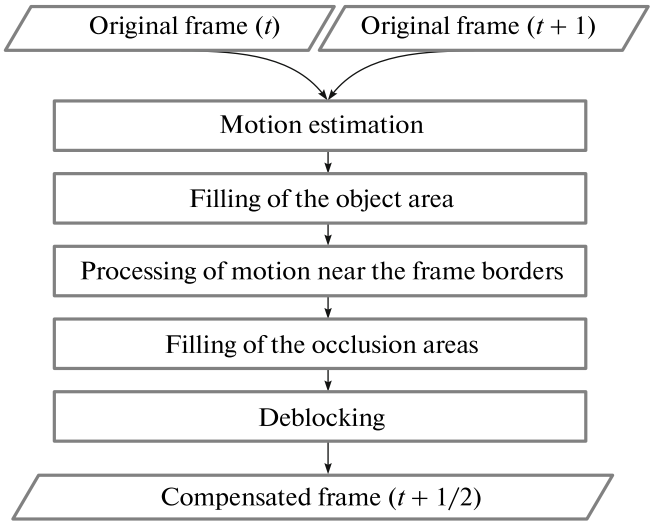

Original frame (t)
Original frame (t + 1)
Motion estimation
Filling of the object area
Processing of motion near the frame borders
Filling of the occlusion areas
Deblocking
Compensated frame (t + 1/2)

Fig. 9. Compensated frame construction scheme.

1. Based on the motion vector of the M–block, we calculate the coordinates of the corresponding 4×4 block in the CF (the coordinates of the block are the coordinates of its left bottom corner). We will call it C-block (compensated block). Both coordinates of this block are divisible by 4.

2. For the resulting C-block, we calculate the coordinates of the corresponding blocks in the MKF and RKF (compensated main, CM–block, and compensated reference, CR–block). Thus, the coordinates of three blocks are now available: C-, CR- and CM-blocks.

3. Brightness values of the pixels of the C-block are calculated on the basis of the brightness values for the CR- and CM-blocks. Value for each pixel of the C-block is calculated as the mean of the values of the pixels located at the same positions in the CR- and CM-blocks.

Consider the coordinate calculation in the C-, CM-and CR-blocks in more detail (see Fig. 10). For every M-block with the coordinates B, the coordinates B_C of the corresponding C-block are calculated by to the following formula (here and below we assume that the abscissa axis is directed from the left to the right and the ordinate axis is directed from the bottom to the top)

$$B_C = \left]\frac{p_0 -]v/2[}{4}\right[\cdot 4, \quad p_0 = B_M + (1, 1), \qquad (6)$$

where

$]\cdot[$ is the rounding operation;

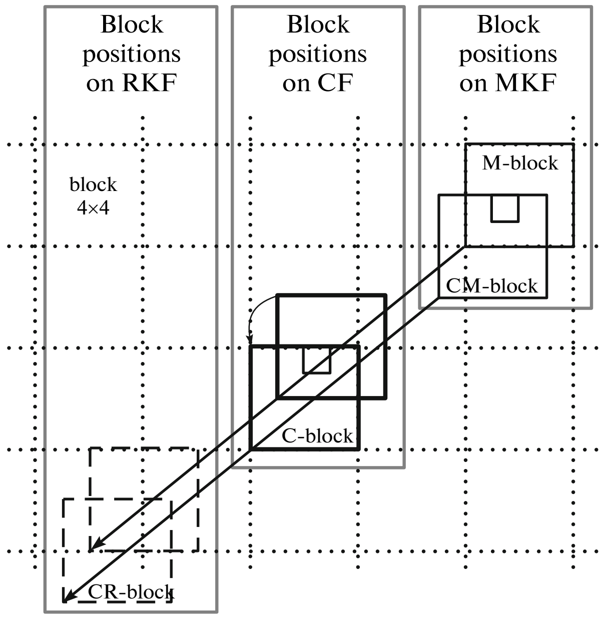
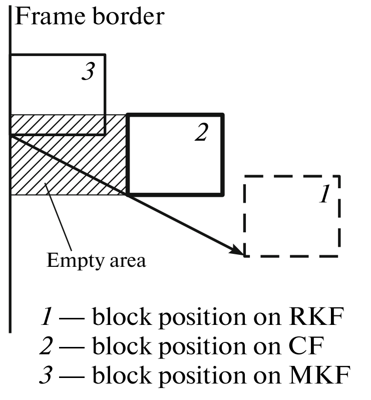


Fig. 10. Filing of the object area.



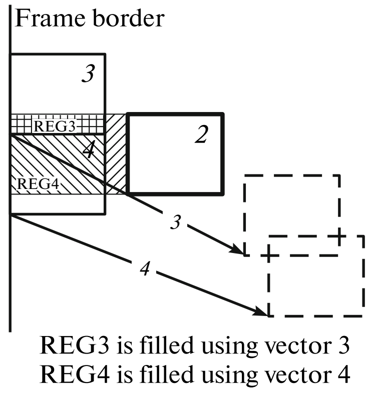


Fig. 11. Method for motion processing near the frame borders.

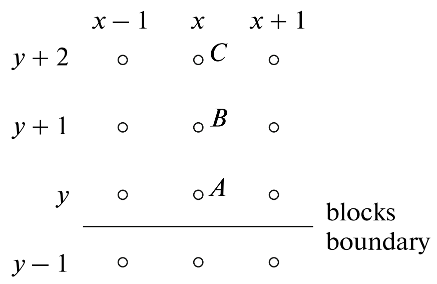


Fig. 12. Neighborhood of the horizontal boundary between blocks.

B_M, B_C are the coordinate vectors of the M-block and the C-block, respectively;

p_0 are the coordinates of the reference point of the C-block;

v is the motion vector obtained for the M-block at the ME stage.

Then, the coordinates of the CM- and CR-blocks are calculated by the formula (see Fig. 10)

$$B_{CR} = B_C + \frac{V}{2}, \quad B_{CM} = B_C - \frac{V}{2}. \tag{7}$$

It is important to note that the coordinates in these formulas are calculated without the operation of taking the integer part. In case when the calculated values of the block coordinates have nonzero fractional parts, the information about the pixel brightness in this block is calculated on the basis of the bilinear interpolation of the brightness values in the frame containing this block.

According to this method of vector calculation in the C-blocks, some parts of the blocks can be compensated with a wrong motion vector; however, this inaccuracy will be taken into account when calculating the pixel error of the compensated frame.

The operation described in this subsection (alignment of the blocks of the compensated frame to the 4×4 mesh) considerably facilitates deblocking at the last step of the the compensated frame construction.

2.2. Processing the Motion on the Frame Borders

The case of the motion nonparallel to the frame border near its boundary can be processed by the ME procedure only in one direction (either forward, or backward). For example, the case when a texture moves towards the image border can be correctly processed only by the forward ME because the border blocks in the current frame are fully visible in preceding frame and are only partially visible in the next frame (if the motion rate is high, they are completely invisible in the next frame). Below, we consider only the motion towards the image border. In this case, the forward ME procedure calculates correct vectors. At the time corresponding to the CF, the moving object is located between the image border and object position in the RKF. Thus, an empty area appears near the border in the CF. Each pixel of this area is compensated using the motion vector of the pixel with the same position in the MKF. This method for determining the vectors for the empty areas is not perfect; however, it makes is possible to process the motion near the frame edges.

A scheme explaining the idea of the process just described and the method for filling empty areas near the frame borders is presented in Fig. 11.

2.3. Correction of the Filled Areas

After the stages described above have been executed, two types of empty blocks can appear on the compensated frame:

1. Blocks that are a part of a moving object or texture;

2. Blocks that are a part of the occlusion areas appearing, for example, near the edges of the moving objects.

At this stage, we fill the blocks of the first type. In order to distinguish between the blocks of different types, the variance is calculated for every coordinate of the motion vector in the 3×3 neighborhood around each empty block. If both obtained values are less than a threshold, the current block is considered to belong to the inner area of the object or texture (block of type 1); its vector is calculated as the arithmetic mean of the vectors of the adjacet blocks. The resulting vector is used to compensate the empty block.

Thus, after this step is completed, only occlusion areas are left empty on the compensated frame.

2.4. Filling the Occlusion Areas

Occlusion areas can be of one of the following types:

1. **Uncovered areas.** These areas appear on the compensated frame when some blocks of the MKF are not seen on the RKF. Such areas appear, for example, near the right edge of an object moving to the left in a fixed background.

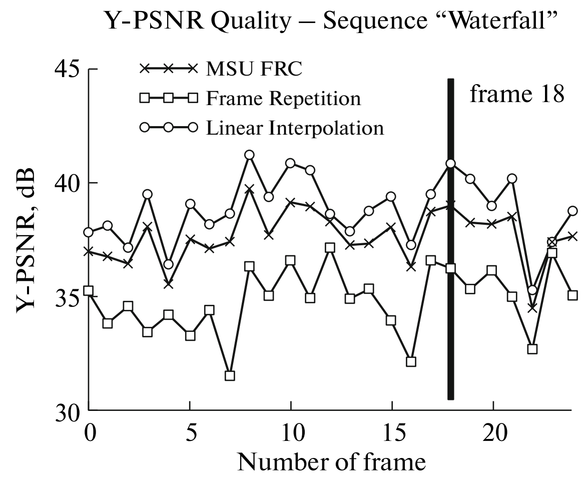


Fig. 13. Y–PSNR plot for the converted sequence *waterfall*.

2. **Covered areas.** These areas appear on the compensated frame when some blocks of the RKF are not seen on the MKF. Similar to the previous case, occlusions of this type appear near the left edge of an object moving to the right in a fixed background.

At the first stage of the CF construction (Section 2.1) blocks with large values of the compensation error are marked as a part of an uncovered occlusion area (uncovered blocks). All other empty blocks are considered to belong to the covered occlusion areas (covered blocks). The uncovered blocks are filled by copying the blocks with the same coordinates in the MKF. Analogously, the covered blocks are copied from the RKF.

Such a scheme of the occlusion area processing is obviously not perfect. It assumes that one of the occlusion objects is static (this is why copying instead of the motion compensation is performed). This scheme works incorrectly, for example, in the case when occlusion appears on the boundary of two moving objects. However, occlusions of this type are quite rare and the proposed scheme makes it possible to process a considerable number of occlusions.

2.5. Deblocking

At this stage, we process the borders of the 4 × 4 blocks where the blocking effect is observed and the pixel compensation errors are quite high (information about the pixel compensation errors is contained in the array of pixel errors of the compensated frame).

The deblocking procedure is performed in two steps; at each step, the block borders are processed on

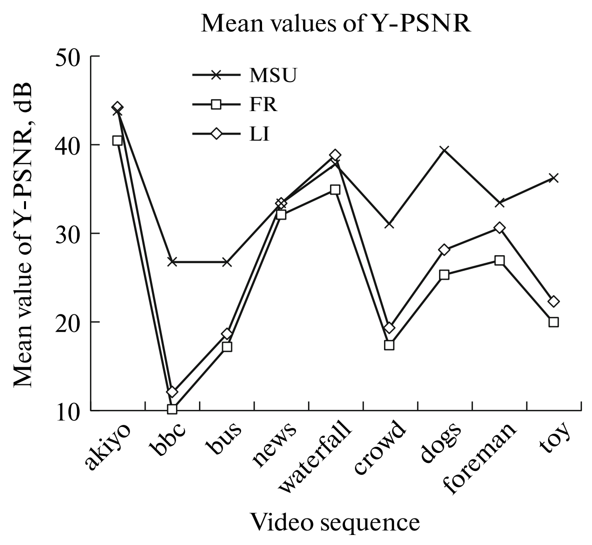


Fig. 14. Plot of the mean Y–PSNR values for all sequences.

Table 1. Y–PSNR (dB) values for all sequences

Sequence	MSU	FR	LI
akiyo	43.8	40.2	44.1
bbc	26.6	10.1	12.1
bus	26.8	17.0	18.7
news	33.2	31.9	33.2
waterfall	37.6	34.7	38.7
crowd	30.9	17.3	19.3
dogs	39.1	25.3	28.1
foreman	33.2	26.8	30.4
toy	36.1	19.9	22.3
Mean	34.1	24.8	27.4

Table 2. Difference in the Y–PSNR (dB) averaged values compared to the FR–method for all sequences

Sequence	MSU–FR	LI–FR
akiyo	3.5	3.8
bbc	16.5	1.9
bus	9.7	1.7
news	1.3	1.3
waterfall	2.8	3.9
crowd	13.6	2.1
dogs	13.7	2.7
foreman	6.3	3.6
toy	16.2	2.4
Mean	9.3	2.6

a 8 × 8 mesh. The mesh used at the second step is shifted by 4 pixels vertically and horizontally relative to the mesh used at the first step. Thus, a 4 × 4 mesh is processed in two steps.

Each step consists of two runs: deblockintg along the horizontal and the vertical borders. The results obtained at the first step are used as the initial data for the second step.

Let us consider processing of the horizontal borders of the blocks. The initial data is the part of the compensated frame I_C of size 3 × 6 pixels (see Fig. 12). The current pixel has the coordinates (x, y); the time coordinate t is omitted to shorten the description. Let us describe how the points with ordinate y in the upper block are processed.

These points are processed if the brightness disbalance D on the block border is greater than the threshold TH_1. The disbalance D is calculated by the formula (see [8] for details)

$$D = |d_1 + d_2 + d_3|, \quad d = c - \frac{a+b}{2}, \tag{8}$$

$$\begin{aligned} a &= k - l, \quad c = l - m, \quad b = m - n, \\ k &= (I_C(x-1, y+1), I_C(x, y+1), \\ &\quad I_C(x+1, y+1)), \\ I &= (I_C(x-1, y), I_C(x, y), \\ &\quad I_C(x+1, y+1)), \\ m &= (I_C(x-1, y-1), I_C(x, y-1), \\ &\quad I_C(x+1, y-1)), \\ n &= (I_C(x-1, y-2), I_C(x, y-2), \\ &\quad I_C(x+1, y-2)). \end{aligned} \tag{9}$$

If D is less than the threshold, the new brightness value $\hat{I}(x, y)$ at the point A (see Fig. 12) is calculated by the formula

$$\hat{I}_C(x, y) = I_C(x, y) + \frac{2}{3} \cdot (-c_2) \cdot q, \tag{10}$$

$$q = \frac{\bar{E}(x, y)}{\bar{E}(x, y-1) + \bar{E}(x, y) + 1}, \tag{11}$$

where

$\bar{E}(x, y)$ is the mean of the pixel compensation error in the neighborhood of the point (x, y) calculated inside the block containing (x, y) and

q is a coefficient depending on ratio of the compensation errors of the adjacent blocks.

The greater the block compensation error compared with the adjacent block, the greater is the degree of processing in this block.

If the module of the difference $|a_2|$ between the brightness of the points A and B (see Fig. 12) is less than the threshold TH_2, then the brightness at the point $(x, y + 1)$ is changed. The new brightness at this point is calculated by the formula

$$\hat{I}_C(x, y+1) = I_C(x, y+1) + \frac{1}{2}(-c_2)q. \tag{12}$$

Analogously, if the module of the difference between the brightness of the points B and C is less than the threshold TH_2, then the brightness of point C is calculated by the formula

$$\hat{I}_C(x, y+2) = I_C(x, y+2) + \frac{1}{4} \cdot (-c_2) \cdot q. \qquad (13)$$

The lower block is processed analogously.

The level of blockiness is considerably reduced at this stage, and image details remain unchanged (due to the use of the threshold TH_2).

2.6. Construction of the Interpolated Frame

After both compensated frames have been calculated, the interpolated frame is constructed. The brightness of each point of the IF is set to the brightness of the pixel with the same coordinates in one of the CFs. The choice of the CF depends on the mean value of the pixel compensation error in the 5 × 5 neighborhood centered at the current point. The CF with the lower mean value of the pixel compensation error is taken.

3. RESULTS

Several standard video sequences were used to compare the methods. These are the sequences with weak motion ("akiyo", "news", and "waterfall"), and the sequences with more complicated motion (all the rest). The following notation is used in the plots and figures: *MSU* is the proposed method, *LI* is the linear interpolation method, and *FR* is the frame repetition method.

The peak signal to noise ratio (PSNR) function evaluated using the brightness component (Y—PSNR) is used as the criterion for the image comparison. The value of this function for two images X and Y of size $N \times N$ is calculated by the formula

$$PSNR(X, Y) = 10\log_{10}\frac{255^2 N^2}{\sum_{i=1}^{N}\sum_{j=1}^{N}(X(i,j) - Y(i,j))^2}.$$

This function is usually called metric in the literature devoted to video processing although, strictly speaking, it is not a metric because it does not satisfy the triangle inequality. It is easy to see that the values of this function are inversely proportional to the actual distance between the images, i.e. the more "similar" the images, the greater is the value of *PSNR*. This function is not perfect for the image comparison; it has disadvantages, which is proved by the examples presented in this section. Nevertheless, it is most frequently used in the papers devoted to video processing. This is why this function was chosen as a comparison instrument in the present paper.

The results of the comparison are presented on Figs. 13, 14 and in Tables 1 and 2. *PSNR* calculated for the initial sequence and for the sequence obtained by replacing every second frame by the interpolated frame. Such comparison makes it possible to estimate the difference between the initial and the interpolated frames using an independent criterion. The values of *PSNR* were calculated only for the interpolated frames.

Analyzing the plots of *PSNR*, one should take into account that the decrease of *PSNR* is often caused by the artifacts that are practically unnoticeable; i.e. small values of *PSNR* do not necessary imply the drop in the visual quality. For example, on frame 18 of the "waterfall" sequence, the *PSNR* value for the proposed method is less than the value given by the linear interpolation method (see Fig. 13); however, the visual quality of the latter method is lower; indeed, the area of the waterfall is blurred in the direction of the water flow motion (see Fig. 15). Analyzing positions of the water flow particles relative to the base lines on different frames in Fig. 15, one can easily see that the wave crests have from the same positions in the frame $(t - 1)$ and $FR(t)$. At the same time, comparing the frames $(t - 1)$, $MSU(t)$, and $(t + 1)$ one can see that the wave crests are at different positions, which shows the smoothness of the motion (particularly comparing to the result obtained using *FR*). The same comparison methodology should be used for the analysis of the results presented in Fig. 16.

The plot in Fig. 14 is obtained by averaging the *PSNR* values over all the interpolated frames of the sequence. The exact mean values for all the sequences and the mean *PSNR* value over all sequences are summed up in Table 1. Table 2 contains the values of the difference between the mean *PSNR* values compared to the *FR* method.

In Fig. 16, one can see the result of the visual comparison of the proposed method (*MSU*) and the *UCSD* method based on the application of the global motion compensation procedure [5]. The motion in the video sequence obtained by the proposed method is smoother compared to the motion in the *UCSD* sequence (Fig. 16). Analyzing the positions of the objects relative to the base lines (special attention should be paid to the motion of feet and heads) one can conclude that the motion in the *UCSD* sequence is jerkier. For the convenience of the comparison of each pair of frames *MSU* and *UCSD*, *PSNR* is visualized on the right. The brighter the area in the visualization frame, the greater is the difference between the frames in the corresponding area.

Since the proposed method does not involve complicated mathematical calculations, video with the resolution of 704 × 576 can processed in real time.

The scope of this method is restricted to the frame rate up-conversions with coefficients that are multiples of two. The most important application of the

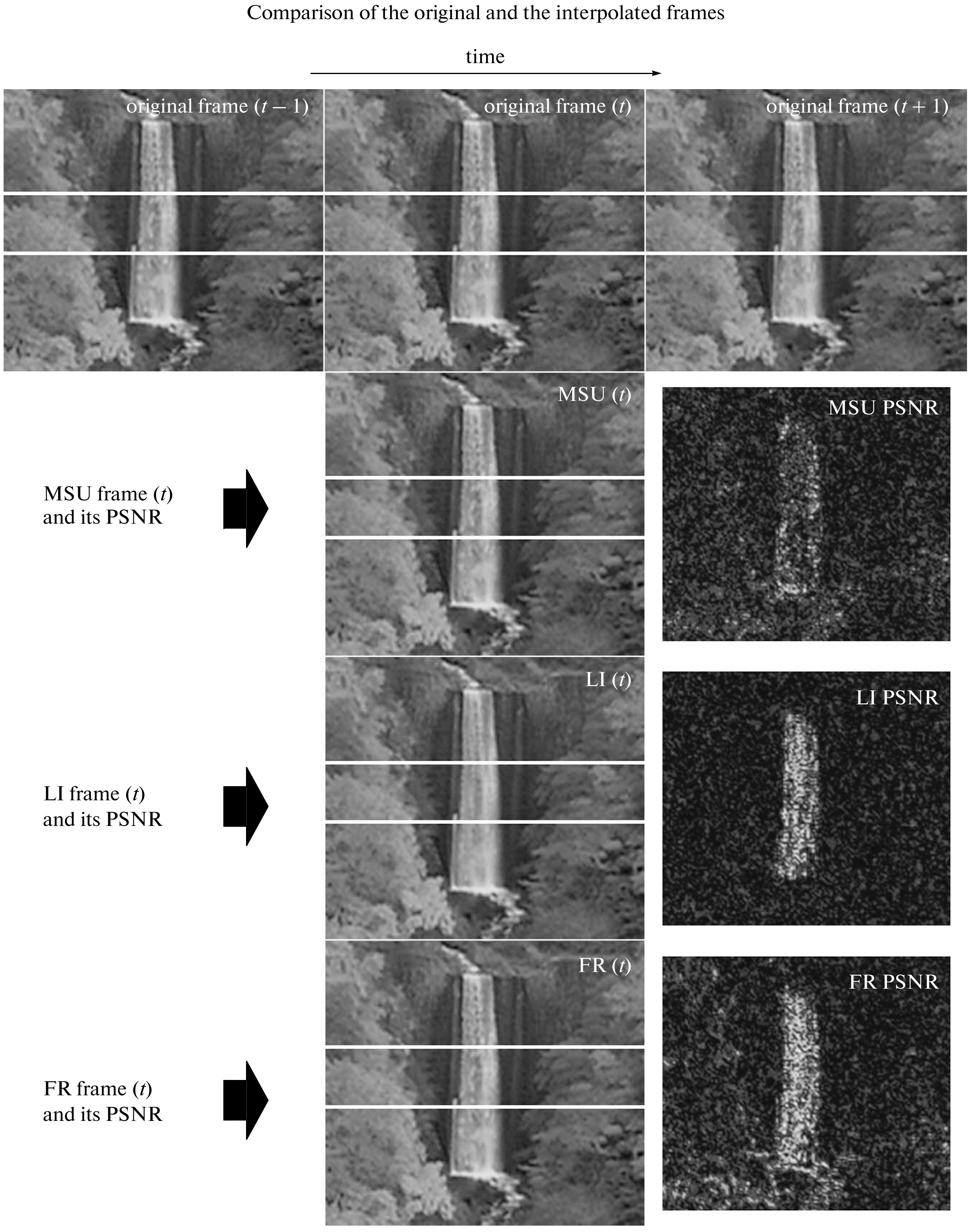

Comparison of the original and the interpolated frames
time
original frame ($t-1$)
original frame (t)
original frame ($t+1$)
MSU (t)
MSU PSNR
MSU frame (t) and its PSNR
LI (t)
LI PSNR
LI frame (t) and its PSNR
FR (t)
FR PSNR
FR frame (t) and its PSNR

Fig. 15. Example the Waterfall sequence processing (frame 18). The right part shows PSNR visualization demonstrating the difference between the original frame and the frame obtained using one of the methods; brightness of each point in the visualization frame is proportional to the error at this point.

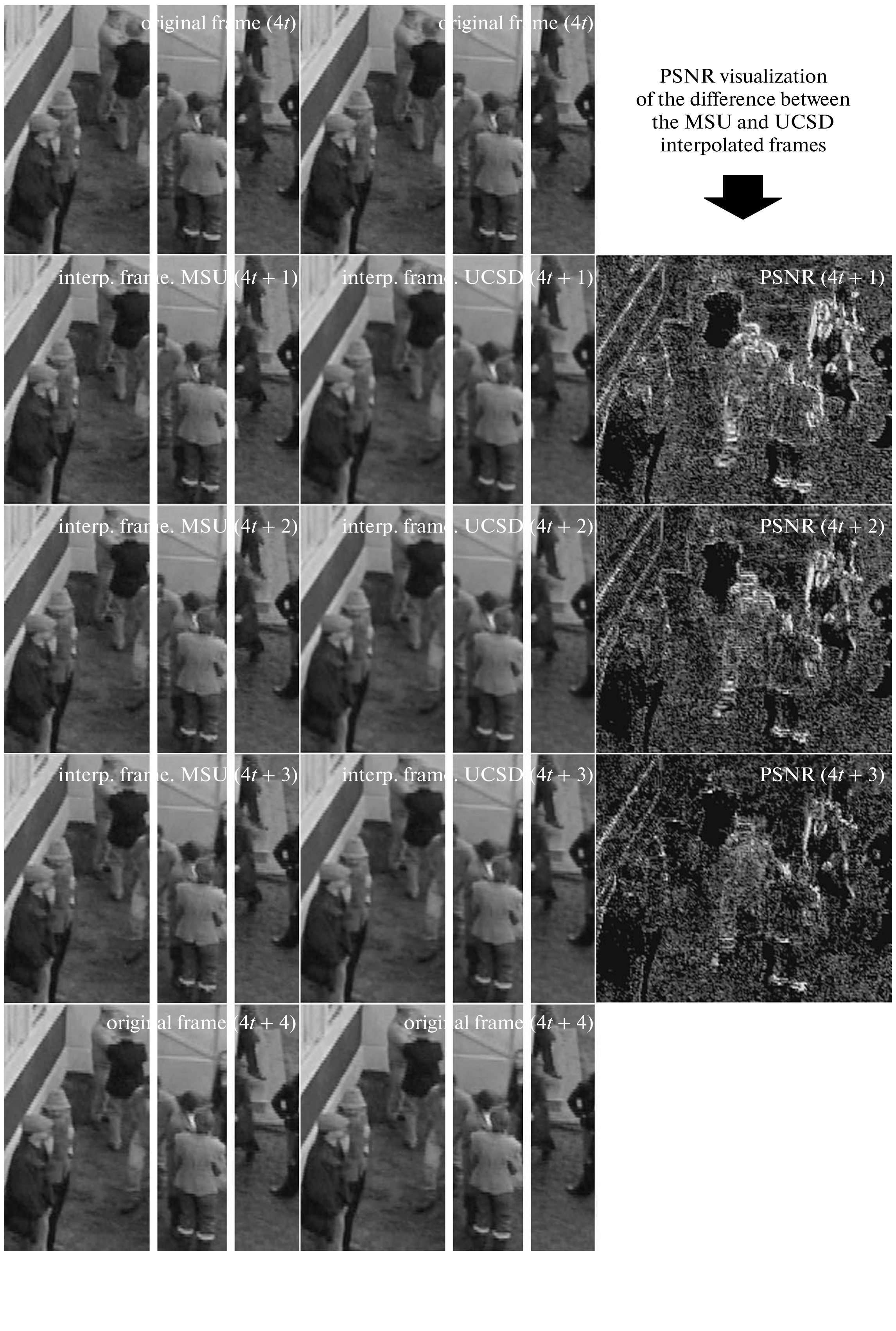

original frame (4t)
original frame (4t)
PSNR visualization of the difference between the MSU and UCSD interpolated frames
interp. frame. MSU (4t + 1)
interp. frame. UCSD (4t + 1)
PSNR (4t + 1)
interp. frame. MSU (4t + 2)
interp. frame. UCSD (4t + 2)
PSNR (4t + 2)
interp. frame. MSU (4t + 3)
interp. frame. UCSD (4t + 3)
PSNR (4t + 3)
original frame (4t + 4)
original frame (4t + 4)

Fig. 16. Visual comparison of MSU and UCSD.

proposed method is its in TV sets with the vertical scan rate of 100 Hz.

4. CONCLUSIONS

The proposed algorithm for the frame rate conversion makes it possible to obtain video sequences with a time-independent sharpness level, which is not typical for other FRC algorithms. The comparison results with other FRC methods showed that the proposed method outperforms the other methods by about 9 dB on the average. The method does not involve any complicated mathematical calculations, which makes it possible to use it real-time video processing.

REFERENCES

1. Castagno, R., Haavisto, P., and Ramponi, G., A Method for Motion Adaptive Frame Rate Up-Conversion, *IEEE Trans. Circuits Syst. Video Technology,* 1996, vol. 6, no. 5, pp. 426–435.
2. Kuo, T.-Y., Kim, J., and Kuo, C.-C. J., Motion-Compensated Frame Interpolation Scheme for H.263 Codec, *Proc. ISCAS 99*, Orlando, 1999, vol. 4, pp. 491–494.
3. Antunez, E., Low-Cost Temporal Interpolation of Video Frames, *Image Communication II*, Class Project, Stanford Center for Image Systems Engineering, 2002.
4. Liu, S., Kim, J., and Kuo, C.-C. J., Non-Linear Motion-Compensated Interpolation for Low Bit Rate Video, *SPIE Proc. Int. Symp. Optical Sci. Engineering and Instrumentation. Applications of Digital Image Processing XXIII*, San Diego, 2000, vol. 4115, pp. 203–213.
5. Kumar, E., Biswas, M., and Nguyen, T.Q., Global Motion Estimation in Frequency and Spatial Domain, *IEEE Int. Conf. Speech, Acoustics, Signal Processing*, Montreal, 2004, vol. 3, no. 17-21, pp. 333–336.
6. Krishnamurthy, R., Woods, J. M., and Moulin, P., Frame Interpolation and Bidirectional Prediction of Video Using Compactly-Encoded Optical Flow Fields and Label Fields, *IEEE Trans. Circuits Syst. Video Technology*, 1999, vol. 9, no. 5, pp. 713–726.
7. Dufaux, F. and Moscheni, F., Motion Estimation Techniques for Digital TV: A Review and a New Contribution, *Proc. IEEE*, 1995, vol. 83, no. 6, pp. 858–876.
8. Vatolin, D. and Grishin, S., High-quality Video Deblocking Method without the Use of Quantization Parameters, *Trudy konferentsii Graphicon-2004* (Proc. Conf. Graphicon-2004), Moscow, 2004, pp. 257–260.