Frame rate conversion based on spatio-temporal smoothness constrained multilayer motion estimation and inpainting

Vikas Ramachandra*, Aimei Huang and Truong Q. Nguyen *Fellow, IEEE*

Abstract

We explore a new technique for video frame rate up-conversion. A noniterative multilayer motion estimation algorithm is investigated, based on spatio-temporal smoothness constraints. For regions in the interpolated frame which cannot be motion compensated, we use an exemplar based video inpainting algorithm. The proposed approach yields excellent results compared to other previous approaches.

EDICS Category: MDE-OTHR

* Corresponding Author

This work was supported by NVIDIA.

The authors are with the University of California, San Diego, 9500 Gilman Drive, San Diego CA 92122. Email: {vikas, aihuang, nguyent}@ucsd.edu. Phone: 858 534 5669.

Frame rate conversion based on spatio-temporal smoothness constrained multilayer motion estimation and inpainting

I. Introduction

Frame rate conversion (FRUC) is essential for several applications [1] including transcoding betweeen video coding standards, camera view synthesis, 120Hz LCDs, intermediate image synthesis for slow-motion effects and conversion of 24 Hz film video to 50 Hz/ 60 Hz television video. An important requirement for FRUC is the estimation of true motion between successive frames of the input video sequence. The motion field estimated should be consistent both spatially and temporally, as well as maintain the structure of the objects in the interpolated frames.

Estimating the motion field can be cast as an optimization problem wherein one tries to minimize an energy function subject to some smoothness constraints. It was shown in [2] that for local smoothness constrained multiple motion estimation, the resulting problem is non-convex, which is solved by multidimensional stochastic sampling.

Although this technique converges to a globally optimal solution, it is too slow for real time applications. There has been some previous works in approximating this cost function with a piecewise supported cost function and finding a locally optimal motion model, and our work is also in this spirit.

Previous work to tackle multiple motion estimation uses a layered approach, wherein the scene is segmented into multiple layers. Each layer is assumed to contain object(s), not necessarily connected, which have the same motion. An affine model is fit to each of these layers [3], [4]. We extend this idea by removing the requirement of a single parameter set for each layer. Instead, by imposing neighborhood smoothness regularization, we allow the local motion in each layer to vary across a constrained range of values. This extends the flexibility of the motion model to include arbitrary local motion.

Simultaneous segmentation of a scene into regions with different motions and estimation of these motions is an ill-posed problem, since the two steps are dependent. To find the best motion for region, the region should be known beforehand. Similarly, in order to find the best segmentation, we need to know the motions for different areas of the scene to start with.

Methods based on the above approach solve this problem recursively [3],[4],[5],[6] leading to convergence to a (locally) best segmentation and motion model after several iterations. Again, this is not easily achievable in real time. We propose to solve this issue by finding an approximate suboptimal solution noniteratively, which nevertheless gives comparably good results.

For FRUC, the intermediate motion compensated reconstructed frames will have 'holes' for some regions which could not be mapped to reference frames. We fill these unmapped regions by using an exemplar based video inpainting technique, which ensures that structures and edges are maintained across frames.

II. Review of previous approaches to motion field estimation and motion segmentation

A. Stochastic sampling

The assumptions about the scenes and images are in terms of 3 constraints described below. The constraints are formalized as energy functions over local neighborhoods or cliques, in a grid. The 3 constraints are:

Data preservation constraint:

This can be related directly to the brightness constancy assumption. The most direct way to use the brightness constancy assumption is to formulate the data conservation error measure using sum of squared difference (SSD) correlation. In this formulation, the image velocity is assumed to be approximately constant within a local neighborhood, R, and the error associated with a given displacement is formulated as:

$$E_D(u,v) = sum_{x,y\in\mathbb{R}}[I(x,y,t) - I(x+\delta t, y+\delta t, t+\delta t)]^2 \tag{1}$$

Here, (u,v) is the image velocity at a point, and δt is a small time change. The above equation simply relates the pixel(s) at location (x,y) at present time t to the best plausible position in a later image, at a location offset by the optical flow.

The support region R must be chosen large enough to avoid the aperture problem, yet small enough to honor the constancy of motion assumption over a local neighborhood.

The aperture problem, in brief, is that using a local window based motion estimation approach, only the motion component which is parallel to the local gradient seen at the window can be inferred. As a result, there will be many true motion vectors, which will satisfy the constraints enforced by the above constraint, making this an ill posed problem.

Therefore, the motion vectors obtained using just the above constraint will be locally 'optimal' with respect to the chosen metric, however, due to the aperture problem, these might not represent the true motion. To regularize this solution, we need to enforce some additional constraints like those enforcing a fixed smoothness variation.

Spatial smoothness preserving constraint:

The simplest formulation for this is to use an energy term which gives the deviation of the motion vector (u,v) at the present location from its neighboring motion vectors in support region S.

$$E_S(u,v) = \sum_{(s)\in S}[(u_s-u)^2 + (v_s-v)^2]^2 \tag{2}$$

Temporal smoothness preserving constraint (E_T):

This is similar to the above term, except that the smoothing is done with respect to local motion vector neighborhoods in adjacent time frames.

The total energy term at a particular location is

$$E_{x,y}(u,v) = E_{D(x,y)}(u,v) + E_{S(x,y)}(u,v) + E_{T(x,y)}(u,v) \tag{3}$$

The overall energy term for the entire image is:

$$E_{global} = \sum_{(x,y)\in MxN} E_{x,y} \tag{4}$$

Instead of finding locally optimal solutions at each location, the problem is posed as estimation of the motion vector set for the full image which gives global energy minimization.

Using the L2 norm makes the problem convex, and can be solved using standard gradient projection techniques, for which off the shelf packages are available. The dimensions of the problem makes the performance slow. The biggest problem with using the L2 norm is that it assumes the motion field to be smooth and continuous, so it performs poorly in the presence of edges and multiple motions within a local neighborhood, as discussed in [2]

Various techniques have been proposed to handle discontinuities resulting due to multiple motions. To allow for discontinuities, a Markov Random Field model (MRF or line process) can be used [2]. However, this makes the energy minimization problem non-convex, requiring a stochastic sampling procedure to obtain the global optimal solution, which is time-consuming and complex.

Another drawback is that for a complicated occlusion, a discontinuity-based segmentation model is inadequate. If an otherwise homogeneous object is partitioned into many disconnected regions on the image plane, an edge-based segmentation model prevents the integration of information across the entire object during the segmentation process, which results in discontinuities across different regions of a motion compensated object. Consider this example, where dots are placed on an otherwise transparent cylinder which rotates around its major axis. Two populations of intermingled random dots are seen, one corresponding to the foreground surface of the cylinder and the other corresponding to the background. When presented with this display, an algorithm that extracts motion information using an edge-based segmentation method will not be able to group the two populations of random dots.

However, both human and animal observers have no difficulty in grouping the two motions. The same phenomena can be found with static imagery: an object viewed through a fence or trees may be partitioned into several disjoint regions on the image plane. The regions of this image could not be grouped together using a line-process or other mechanism that relies solely on an edge map for segmentation.

B. Multilayer motion estimation

To overcome the limitations mentioned above, an alternate approach is to segment the image into different layers, and perform the smoothness constrained flow field estimation for each of these layers [6]. This involves the segmentation of a scene as a set of support maps, each corresponding to a distinct homogeneous (but possibly disconnected) region with similar motion parameters. This avoids smoothing across multiple motions.

Robust estimation methods have become popular for image processing, since they have been found to be tolerant to occlusion and other outlier contamination. The use of a support map for estimation is simply an instance of outlier rejection, which is a well-known robust estimation method. In this approach a confidence factor is used to weight the contribution of each point to the estimation. The confidence value is itself iteratively updated based on the residual error of the current iteration. Formally, this type of estimation is known as M estimation. M-estimators are maximum likelihood estimators which allow an arbitrary error norm. Given an image data vector d, and a model, y(x), we wish to find parameters x which are most likely to have generated the observed data 'd'. This approach uses the hidden data framework, and applies an EM like algorithm. The support maps and the motion parameters for each layer are found alternately and iteratively. At each iteration, the confidence levels of different regions of the support maps are updated based on the model fit error using the multilayer motion parameters. Then, based on the new support map set, a new set of model parameters is fit, and the process is repeated to convergence.

A specific model which has been used earlier is a multilayer affine model, wherein an affine parameter set is fit to each layer using the above method. More general models can be used which give better results with greater computational cost.

Although this approach works fairly well, it has a couple of drawbacks. Firstly, since the technique uses a generative model to explain the data, we need to fit a model to each layer, which is difficult to determine well enough for every new video sequence. Also, none of the models might fit well enough, and it is worse to use a bad model rather than use no model at all (i.e. using some naive approach). Also, this procedure as noted above, is iterative (hence slow), and we noticed that not letting the algorithm run to convergence will result in coarse approximate segmentation/ support maps, which leads to breakage of object boundaries. We propose to find an good alternative non-iterative approximation to this method, and use inpainting to overcome the errors resulting due to the coarse approximation. We will discuss in a later section how holes often result in interpolated frames in regions containing object boundaries and discontinuities, and how inpainting can be used to fill these holes while maintaining object structure at the same time.

C. Connected mesh based motion estimation

In this method, a mesh is overlaid on the image, and the mesh consists of triangular elements. Each triangular patch is fit with a parametric motion model (usually projective or bilinear), given the connectivity preserving constraints with respect to its neighbors [8]. Most motion estimators use block matching techniques, which can account for translational motion only. An extension of this is to match each block with transformed blocks of the search region. These transformations could be affine, projective or bilinear. We can improve this generalized block matching procedure by adding connectivity constraints across the boundaries/nodes of adjacent blocks or patches. Although this method is better than naive block matching and can account for arbitrary local motions, this again does not preserve the edges of objects which are spread across many patches.

III. Overview of existing FRUC techniques

In this section, we review some existing FRUC methods (in the context of 2:1 FRUC for simplicity) and discuss their performance issues as well.

A. Frame repetition

This is a simple method where every other original frame is repeated (in case of 2:1 FRUC). There is no actual interpolation taking place. Although this method is obviously very fast, it suffers from motion jitter and jerkiness because frame repetition does not present a smooth change in the motion of the video sequence to the viewer.

B. Frame averaging

Another simple method is to let the intermediate frames just be the average of the adjacent original frames. This does not work well since it leads to a ghosting effect across boundaries due to the effect of pixels from 2 original frames on the interpolated frame.

For better performance, we need to look at FRUC methods which use motion information, i.e. motion compensated FRUC techniques. Let $f(x,t)$ and $f(x,t+T)$ denote the two known images at time t and t+T, respectively, where $x = [x_1 x_2]^T$ stands for the spatial coordinates and T for the time interval between two successive images. The image to be interpolated is represented by $f(x,t+\tau)$ with $0 < \tau < T$. This is illustrated in figure 1. Three popular motion-compensated algorithms described in the literature are occlusion based FRUC, temporal shift FRUC, and motion compensated bidirectional FRUC algorithms. [16] detailed comparison and description of these algorithms, and the authors show that the occlusion based FRUC does better than the other 2 methods. Hence, we use occlusion based FRUC, which is briefly described next.

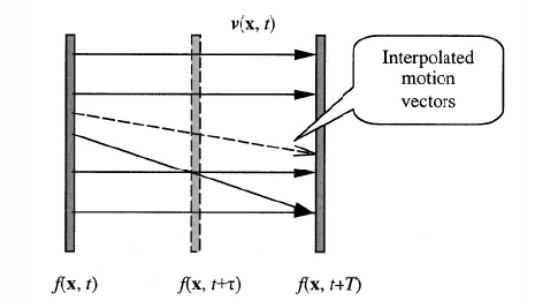


Fig. 1. Motion compensated FRUC

C. Occlusion based FRUC

The unidirectional motion from f (x,t) to f (x,t+T) is estimated using block matching. This results in one motion vector for each block in f (x,t). This motion vector is then assigned to every pixel of the block. Let v(x,t) denote the vector assigned to the pixel at the spatial point x. The motion trajectory passes from point x in f (x,t), through point x + α v(x,t) in the image f (x,t+τ) to be interpolated, to point x+v(x,t) in f (x,t+T), where α = τ/T. It should be noted that point x + α v(x,t) may be located between pixels in the image f (x,t+τ), in which case, the pixels at the closest integer location is assigned value based on that motion vector. Let y = $\lfloor$ x + $\alpha v(x,t)$ $\rfloor$ denote the closest pixel to point x + α v(x,t). The interpolation is performed as:

$$f(y, t+\tau) = (1-\alpha)f(y - \alpha v(x,t), t) + (\alpha)f(y + (1-\alpha)v(x,t), t+T) \tag{5}$$

In this method, we need to handle overlapped motion trajectories and holes due to occlusion [16]. Two different motion trajectories might cross the same pixel in the image to be interpolated. This is caused by motion occlusion, where one object is covered by another one. Generally, the motion trajectory that results in the larger absolute difference is associated with a covered object, while that with the smaller absolute difference is associated with a covering object. The latter is more reliable than the former. Therefore, we use the motion trajectory resulting in the smaller absolute difference for the interpolation.

After all of the motion vectors v(x,t) have been used in equation 5 for the interpolation, there may still be some pixels in f (x,t+τ) that are not interpolated because there is no motion trajectory passing through them. As a simplest method, the values at these locations can be interpolated based on the neighboring pixels, or copied from the reference frame using motion vectors interpolated from the existing motion vectors in the neighborhood. As one would expect, such simplistic techniques to fill in the holes do not work well in practice. Later sections discuss in more detail on how to fill in these holes at object boundaries in a cohesive manner.

It is interesting to note that all these popular FRUC algorithms *choose* to use just block matching, and not the better performing but computationally costly motion estimation methods. There is tradeoff here, where some

performance is sacrificed for computation. However, these FRUC algorithms as such can use any motion estimation method in conjunction with themselves, (not just block matching), as long as the motion estimation procedure is not too intensive.

IV. Proposed algorithm: FRUC using multilayer ME with local neighborhood smoothing

Our proposed algorithm, for every pair of successive video frames, is as follows:

- Segmentation of each video frame into multiple layers.
- Motion estimation for each layer.
- Construction of the intermediate frames using motion information.
- Filling in the gaps in the reconstructed intermediate frames.

Each of the above steps is explained below in detail:

A. Segmentation of each video into multiple layers

An affine model is fit between the successive frames using RANSAC [7]. For this, blocks of fixed size are considered in either image. Once the affine parameters have been determined, for every block, we calculate the error/block resulting due to the affine fit. If this error exceeds a threshold, we label the block as being in the next 'higher' layer. This procedure is repeated until we hierarchically segment the successive frames into required number of layers. One way to stop the segmentation process when the error due to the affine fit for the present layer falls below a predetermined threshold. This is similar to [4] except that our method is non-iterative, and it includes the following local motion refinement step. This could be extended to more general projective and nonparametric models as well. However, due to the approximate non-iterative process, this segmentation is coarse needing further refinement of motion vectors at a local level.

B. Motion estimation for each layer

We estimate the local motion for the blocks for each layer by using block matching with smoothness constraints. This could also be used along with more complex motion segmentation techniques (in the previous step) as a local refinement stage. If required, this can be extended to a more general model at the cost of higher complexity. Let one of the successive frames be the reference frame. For each block in the reference frame which belongs to a given layer, we define an energy function E, spatial smoothness function S, temporal smoothness function T and block intensity match residual error R. These are related as follows:

$$E_{(u,v)} = \alpha_1 * S_{(u,v)} + \alpha_2 * T_{(u,v)} + \alpha_3 * R_{(u,v)} \tag{6}$$

where (u, v) is the image block index for which we have to find the best MV. α_ks are the corresponding weighting factors for each term. Let the block matching function return the top N motion vectors MV_k. Here the performance improves with a larger value of N, especially when the motion vector field is spurious, in which case the true motion vector is farther away from that motion vector which gives the least block matching error. Ideally, we would like to choose from, and hence keep all the motion vectors returned by the block matching function. However, to decrease running time, we keep the top 25 motion vectors, which seems to work well for all our test sequences. We also observe that the performance varies with the order of magnitude of N in the case of an extremely spurious motion vector field returned by block matching.

For each of these motion vectors, the right hand terms are defined as:

$$T_{(u,v,k)} = \sum_{m=u-X_R}^{u+X_R} \sum_{n=v-Y_R}^{v+Y_R} I_{(m,n,l)} * MV_{diff} \tag{7}$$

where

$$MV_{diff} = \{MV_{(u,v,k,t)} - MV_{sel(m,n,t-1)}\}^2 \tag{8}$$

The time indices are specified by t and $t-1$ corresponding to the present intermediate frame being interpolated, and the last interpolated frame. X_R and Y_R are the local search support range values. $I_{(m,n,l)}$ is a binary indicator which is 1 when the index (m, n) belongs to layer $'l'$ and 0 otherwise.

Similarly, for spatial smoothness, we have

$$S_{(u,v,k)} = \sum_{p=u-X_R}^{u} \sum_{q=v-Y_R}^{v} I_{(p,q,l)} * \{MV_{(u,v)} - MV_{sel(p,q)}\}^2 \tag{9}$$

The spatial smoothness support is above and to the left of the present index only, since we do not have the MV values for the indices to the right and bottom of the present index (u, v) when we interpolated for index (u, v). All functions are evaluated over their respective independent local neighborhood supports around the present index (u, v).

The best MV (MV_{sel}) is selected as the MV which gives the least E function over all candidate k MVs for the index (u, v),

$$MV_{sel(u,v)} = argminE_{MV_{(u,v,k)}} \tag{10}$$

This local support based spatio-temporal smoothing is in the spirit of [8], although our approach uses blocks instead of generalized patches. Our model can easily be extended for image arbitrary patches by replacing the

block matching function with a patch fitting function. While the authors in [8] and [9] do not use a multilayer approach, the authors in [4] use only a fixed global model for each of their layers. We believe that combining the 2 approaches of multilayer segmentation with local spatio-temporal smoothing will give the best results. Moreover, unlike [10],[11],[12] and [13], we achieve this using a less complex non-iterative procedure to reach a local solution for the motion field, which also maintains the structure and edges in the video with good spatio-temporal consistency. The tradeoff is that we obtain only an approximate optimal solution. However, this seems to work well in practice as the results show. With respect to review section 2, our method is similar to multilayer ME approach [6], the important differences being that we use a more flexible *local* motion model instead of a fixed parametric model to fit each layer's motion, and also that our procedure is non iterative. The performance of our approximate procedure is improved with the use of inpainting in conjunction with FRUC, as discussed later.

C. Construction of intermediate interpolated frames / FRUC

We use the motion vectors obtained in the previous step. The MVs obtained are for 2 successive frames of the original sequence, say $Frame_{t-1} and Frame_t$. To insert M frames in between every pair of successive frames, we divide the time line into M segments, and increment the MVs by (k/M) for the k^{th} frame we insert (linear interpolation of motion), i.e.

$MV_{(u,v,t+(k/M))} = (k/M) * MV_{(u,v,t)}$

Our method of constructing the interpolated frames extends the occlusion based FRUC technique discussed in section 3.4. Due to the MV subsampling, interpolation and rounding effects, there will be some regions in the intermediate frames which will not be filled in by motion compensated blocks, especially near object separation boundaries and edges. For such regions, we can either apply an order statistic filtering or perform an inpainting. Inpainting is a better option since it avoids blur introduced by filtering. Moreover, inpainting also retains and propagates the structure and edges in the image.

Inpainting helps improve our approximately optimal local solution (which we have obtained in the previous stages) for segmentation and motion estimation. Since we are combining inpainting, which is a dense pixel estimation method, with a block based motion approach, our model can be viewed as a semi-dense model which switches to dense pel estimation only when non-dense block-based approach does not perform well.

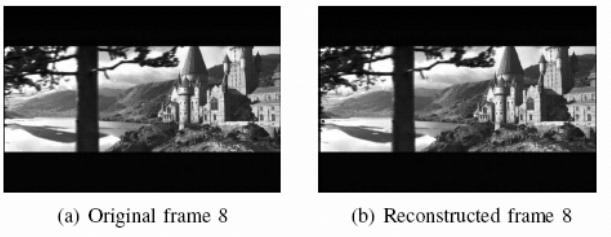
D. Video inpainting to plug the holes

In [14], the authors present a spatio-temporal continuity preserving video inpainting technique. We use the same algorithm here for filling in the holes in the interpolated frames. To our knowledge, this is the first use of video inpainting with FRUC in this manner. This gives better results than using variants of order statistic and median

filtering to get the values of the missing pels. For a comprehensive analysis of various hole filling techniques in general for FRUC, and some inpainting methods in particular, the reader is referred to [17], [18] and [19].

V. Results

We compare the proposed method (Method 1) with 1 layer spatio-temporal smoothing (Method 2 - 'Spatiotemp') [8], a 2 layer affine motion model based motion estimation based FRUC (Method 3 - 'Affine 2') [4], and a postprocessed variant of classification based block matching technique (Method 4 - 'CBM') [15]. We chose 'CBM' as one of the techniques to compare against, because it is representative of techniques which use sliding block method to do FRUC, thereby eliminating the occurrence of holes in the intermediate interpolated images. This, as we shall see, comes at the cost of blur near object boundaries and edges. We present the results here for the 'Castle and tree' sequence also called the 'Harry Potter' sequence. We drop every other (even) frame of the original sequence, and perform 1:2 frame rate upconversion (FRUC). For methods 1 and 3, we use only a 2 layer segmentation for simplicity. We compare the reconstucted even frames with the original frames. As claimed, Method 1 performs better than the rest, both in terms of PSNR as well as maintenance of the structures and object edges and boundaries. Since it is not easy to compare the full frames of all methods, we select different portions of a particular reconstructed frame (Frame 8), and from the figures, it can be seen that Method 1 gives superior performance. Method 2 comes closest but has problems maintaining some of the object boundaries. Method 3 performs poorly over many image regions including but not restricted to boundaries, since we have very few parameters and the method cannot adapt to local variations. Method 4 results in ghosting effects as well as loss in resolution due to the filtering. All methods were combined with video inpainting to fill in the regions which could not be motion compensated. Inpainting performs better than order statistic filtering. The full reconstructed frames for different sequences and videos are available at http://videoprocessing.ucsd.edu/~vikas/FRUC.html



(a) Original frame 8 (b) Reconstructed frame 8

Fig. 2. Original and reconstructed frame of the castle sequence

The castle and tree sequence has the camera panning to the right. So, the objects in the scene move to the left with different velocities, depending on their respective depths from the camera. The motion is almost purely translational. Figure 2 shows frame 8 of the Castle and tree sequence, and frame 8 reconstructed using the proposed

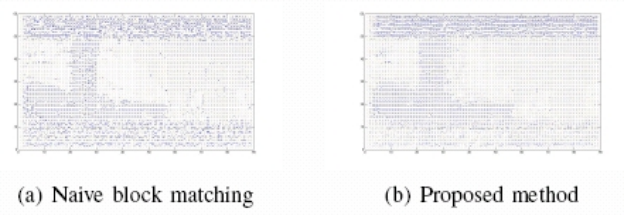


Fig. 3. Motion vector fields- Block matching and proposed

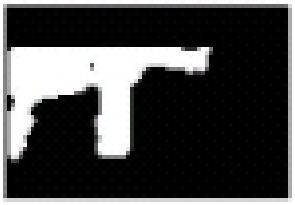


Fig. 4. 2 layer segmentation mask obtained by proposed method

method. It can be seen that the proposed method performs well in predicting the motion, as well as maintaining the structure of all the objects in the scene. The scene can be segmented easily into layers with different motion. An example of this would be Layer 1 containing the background sky and hills, layer 2 containing the castle region, and Layer 3 containing the tree. Figure 3 shows the motion vector field obtained using simple block matching algorithm and the spatiotemporal smoothed field with multilayer segmentation using the proposed method. It can be seen that the spurious motion vector have been well smoothed out. Figure 4 shows one of the segmented layers, namely Layer 3 containing the tree. Figures below give comparisons of the proposed method (Method 1) with methods 2, 3 and 4 respectively.

The remainder of this section has comparisons of Method 1 with methods 2, 3 and 4 for the following well known sequences: Bus, Mobile and calendar, foreman, soccer, football, walk and Formula 1. Regions where Method 1 performs markedly better than other methods are shown here for the sake of brevity.

TABLE I

COMPARISON OF MEAN PSNR VALUES

Proposed method	26.5663
Spatio-temporal smoothing	25.7901
2 layer affine model	19.9766
CBM	25.28

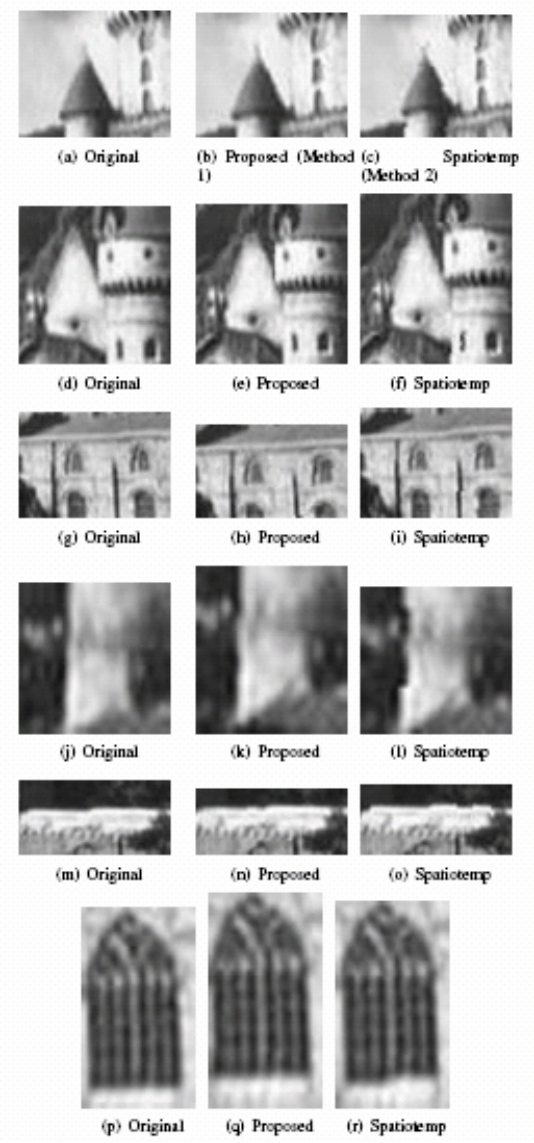


Fig. 5. Comparison of the proposed method with 'spatiotemp' method

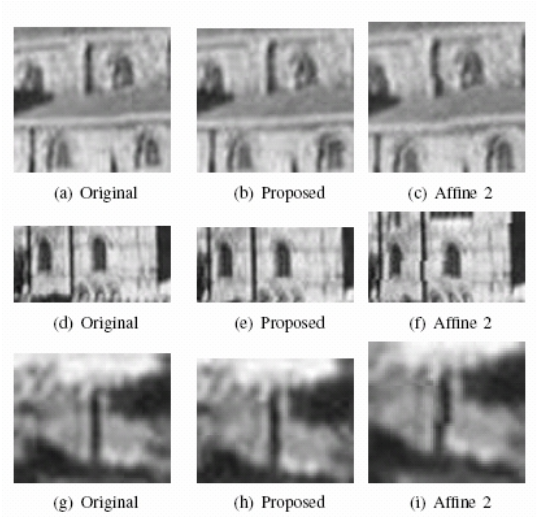


Fig. 6. Comparison of the proposed method with 'affine2' method

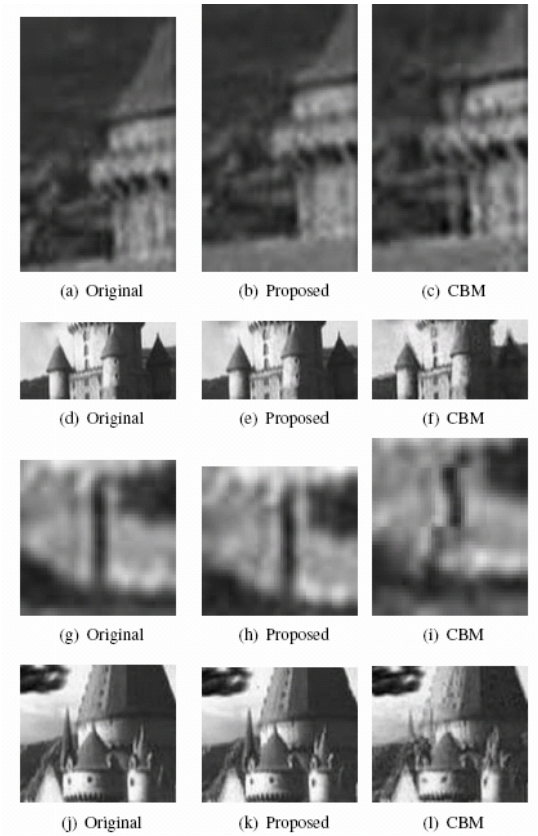


Fig. 7. Comparison of the proposed method with 'CBM' method

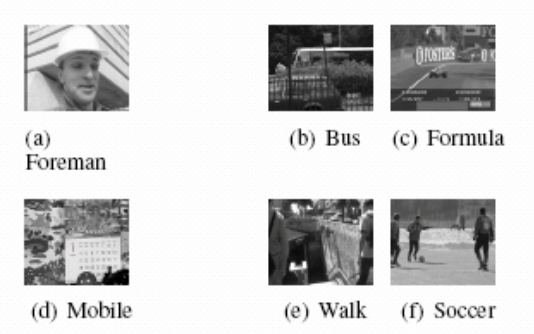


Fig. 8. A frame of sequences used for testing

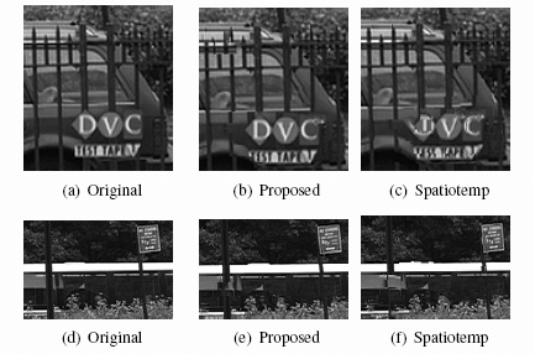


Fig. 9. Comparison of the proposed method with 'spatiotemp' method for BUS sequence

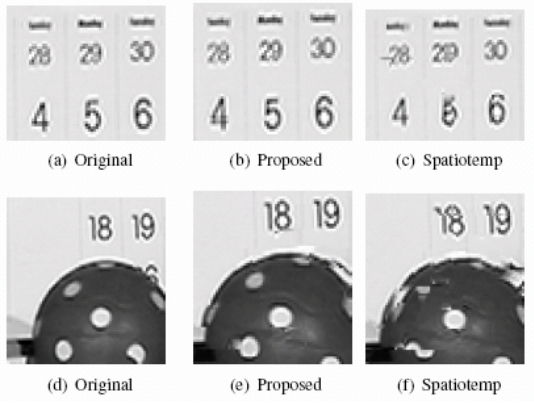


Fig. 10. Comparison of the proposed method with 'spatiotemp' method for MOBILE sequence

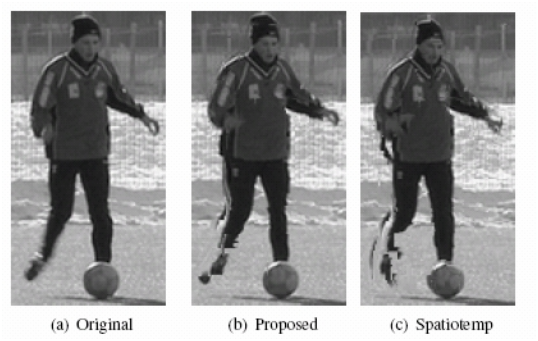


Fig. 11. Comparison of the proposed method with 'spatiotemp' method for SOCCER sequence

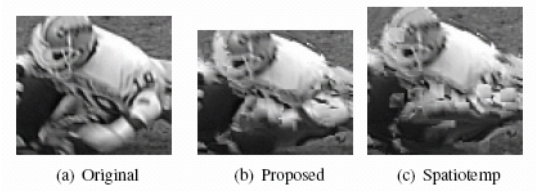


Fig. 12. Comparison of the proposed method with 'spatiotemp' method for FOOTBALL sequence

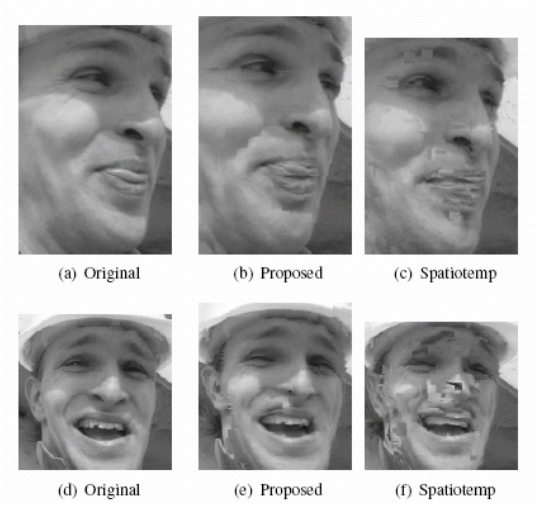


Fig. 13. Comparison of the proposed method with 'spatiotemp' method for FOREMAN sequence

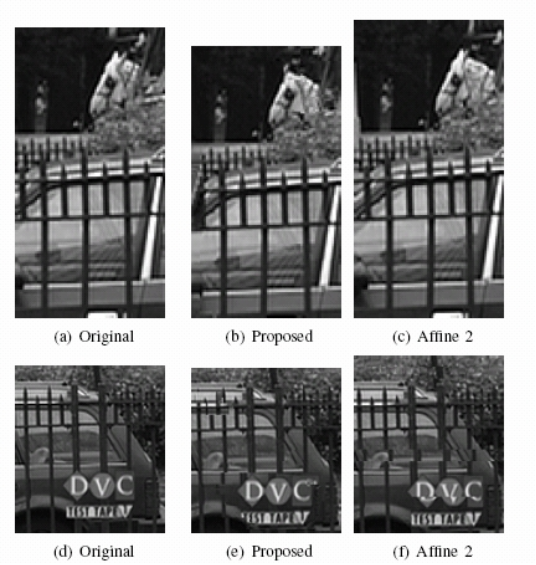


Fig. 14. Comparison of the proposed method with 'affine 2' method for BUS sequence

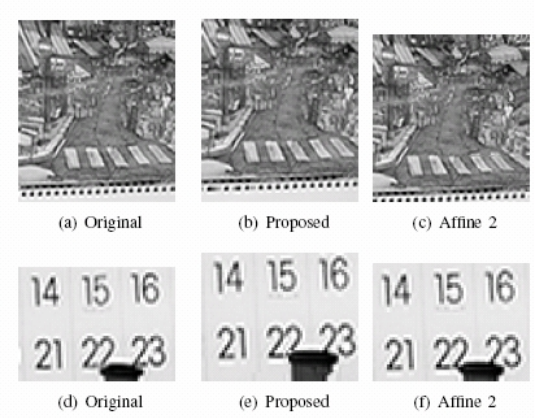


Fig. 15. Comparison of the proposed method with 'affine2' method for MOBILE sequence

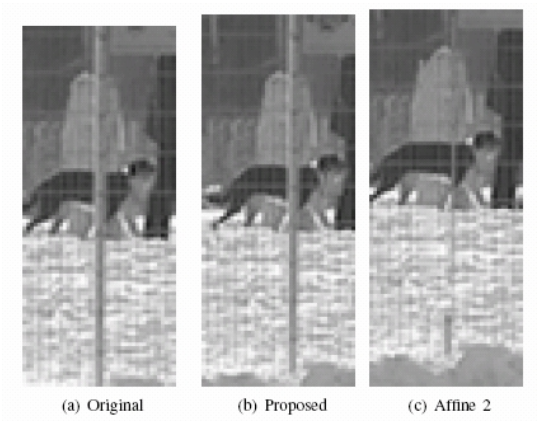


Fig. 16. Comparison of the proposed method with 'affine2' method for SOCCER sequence

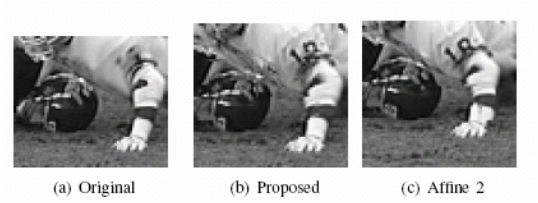


Fig. 17. Comparison of the proposed method with 'affine2' method for FOOTBALL sequence

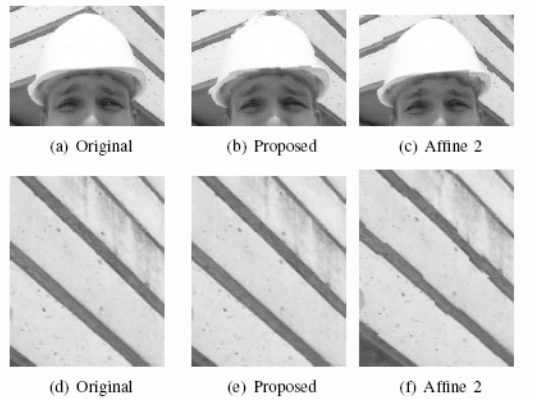


Fig. 18. Comparison of the proposed method with 'affine2' method for FOREMAN sequence

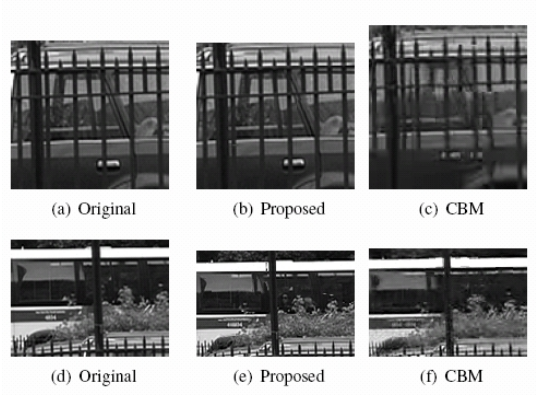


Fig. 19. Comparison of the proposed method with 'CBM' method for BUS sequence

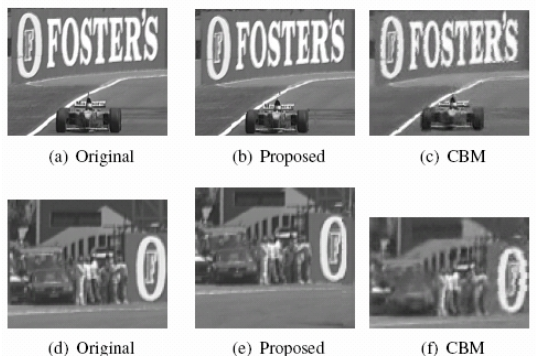


Fig. 20. Comparison of the proposed method with 'CBM' method for FORMULA sequence

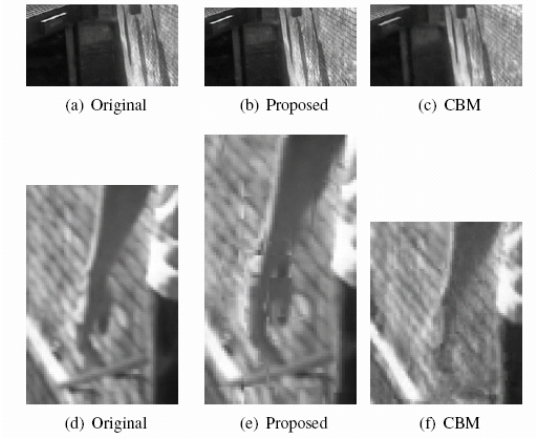


Fig. 21. Comparison of the proposed method with 'CBM' method for WALK sequence

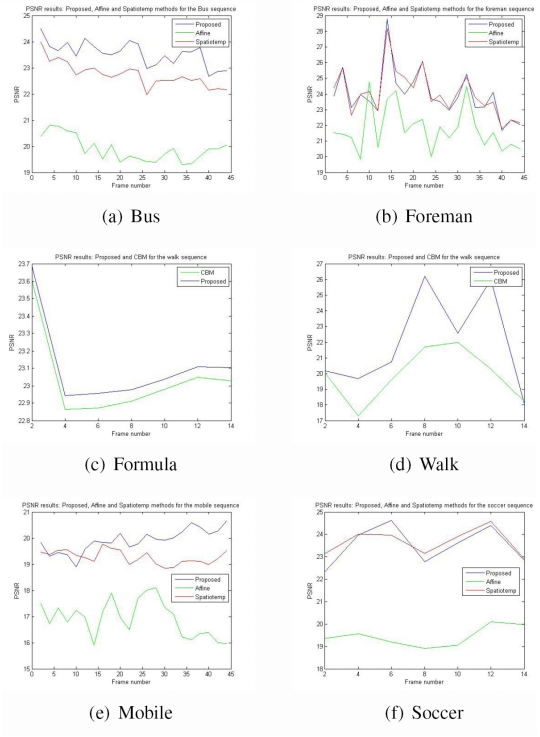


Fig. 22. Comparison of the PSNR of the proposed method with other methods for different sequences considered

VI. Conclusion

We have proposed a frame rate upconversion method based on a noniterative multilayer segmentation, combined with spatio-temporal smoothing at each layer. The method gives an approximately locally optimal motion vector field. The regions which cannot be motion compensated are filled in by inpainting, which preserves the edges and object boundaries better than filtering to fill in the gaps. The method gives good results on the sequences tested (which mostly involve pan shots, translational motion and a set of well visually definable layers), both visually and in terms of PSNR.

References

[1] Filling in the gaps. Collis, B.; Kokaram, A. Electronics Systems and Software, Vol.2, Iss.4, Aug./Sept. 2004 Pages: 22- 28.

[2] Robust dynamic motion estimation over time. Black, M.J.; Anandan, P. Computer Vision and Pattern Recognition, 1991. Proceedings CVPR '91., IEEE Computer Society Conference on, Vol., Iss., 3-6 Jun 1991 Pages:296-302

[3] Layered representation for motion analysis. Wang, J.Y.A.; Adelson, E.H. Computer Vision and Pattern Recognition, 1993. Proceedings CVPR '93., 1993 IEEE Computer Society Conference on, Vol., Iss., 15-17 Jun 1993 Pages:361-366

[4] Layered motion estimation. Schutten R. J.; Pelagotti A; De Haan,G, Philips Journal of Research, Vol 51, No. 2, 1998. Pages:253-267.

[5] Simultaneous motion parameter estimation and image segmentation using the EM algorithm. Matthews, K.E.; Namazi, N.M.; Image Processing, 1995. Proceedings., International Conference on Volume 1, 23-26 Oct. 1995 Page(s):542 - 545 vol.1

[6] Robust estimation of a multi-layered motion representation. Darrell, T.; Pentland, A. Visual Motion, 1991., Proceedings of the IEEE Workshop on, Vol., Iss., 7-9 Oct 1991 Pages:173-178

[7] Global motion estimation in frequency and spatial domain. Kumar, S.; Biswas, M.; Nguyen, T.Q. Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on, Vol.3, Iss., 17-21 May 2004 Pages: iii- 333-6 vol.3

[8] Closed-form connectivity-preserving solutions for motion compensation using 2-D meshes. Altunbasak, Y.; Tekalp, A.M. Image Processing, IEEE Transactions on, Vol.6, Iss.9, Sep 1997 Pages:1255-1269

[9] Multiwindow least-square approach to the estimation of optical flow with discontinuities. F. Bartolini, V. Cappellini, C. Colombo, and A. Mecocci. Opt. Eng., 32(6):1250–1256, 1993.

[10] Optimizing Motion Estimation with Linear Programming and Detail-Preserving Variational Method. Hao Jiang, Ze-Nian Li, Mark S. Drew. CVPR, pp. 738-745, 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04) - Volume 1, 2004

[11] Simultaneous Inpainting and Motion Estimation of Highly Degraded Video-Sequences. Cocquerez, J. P.; Chanas, Laurent; Talon, B. J., SCIA 2003, LNCS 2749, Pages: 685-692.

[12] Hierarchical Estimation and Segmentation of Dense Motion Fields. Memin, E; perez, P. IJCV 46(2), 2002, Pages: 129-155.

[13] Edge-preserving regularization of disparity and motion fields. Yang,W.; Ngan, K.N.; Lim, J.; Sohn, K. Video/Image Processing and Multimedia Communications, 2003. 4th EURASIP Conference, Vol.1, Iss., 2-5 July 2003 Pages: 71- 76 vol.1

[14] Spatio-temporal texture synthesis and image inpainting for video applications. Kumar, S.; Biswas, M.; Belongie, S.J.; Nguyen, T.Q. Image Processing, 2005. ICIP 2005. IEEE International Conference on, Vol.2, Iss., 11-14 Sept. 2005 Pages: II- 85-89.

[15] A Novel Motion Compensated Frame Interpolation Based on Block-Merging and Residual Energy. Ai-Mei Huang; Truong Nguyen Multimedia Signal Processing, 2006 IEEE 8th Workshop on, Vol., Iss., Oct. 2006 Pages:395-398

[16] Comparison of motion-compensated algorithms for frame interpolation; Demin Wang, Opt. Eng. 42, 586 (2003)

[17] Fields of Experts: a framework for learning image priors. Roth, S.; Black, M.J. Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, Vol.2, Iss., 20-25 June 2005 Pages: 860- 867 vol. 2

[18] Region filling and object removal by exemplar-based image inpainting. Criminisi, A.; Perez, P.; Toyama, K. Image Processing, IEEE Transactions on, Vol.13, Iss.9, Sept. 2004 Pages: 1200- 1212

[19] Fast Image Inpainting Based on Coherence Transport.Bornemann F; T. Mrz. Preprint, 28pp, Technische Universitt Mnchen (March 2006). (Forthcoming in Journal of Mathematical Imaging and Vision.)