Temporally Consistent High Frame-Rate Upsampling with Motion Sparsification

Dominic Rüfenacht and David Taubman
Interactive Visual Media Processing Lab (IVMP)
School of Electrical Engineering and Telecommunications, UNSW, Sydney, Australia
{*d.ruefenacht, d.taubman*}*@unsw.edu.au*

***Abstract*—This paper continues our work on occlusion-aware temporal frame interpolation (TFI) that employs piecewise-smooth motion with sharp motion boundaries. In this work, we propose a triangular mesh sparsification algorithm, which allows to trade off computational complexity with reconstruction quality. Furthermore, we propose a method to create a background motion layer in regions that get disoccluded between the two reference frames, which is used to get temporally consistent interpolations among frames interpolated between the two reference frames. Experimental results on a large data set show the proposed mesh sparsification is able to reduce the processing time by 75%, with a minor drop in PSNR of 0.02 dB. The proposed TFI scheme outperforms various state-of-the-art TFI methods in terms of quality of the interpolated frames, while having the lowest processing times. Further experiments on challenging synthetic sequences highlight the temporal consistency in traditionally difficult regions of disocclusion.**

I. Introduction

Video capture and display technology has improved considerably over the last years [1], which creates new challenges and opportunities for video compression and temporal frame interpolation (TFI) algorithms. For example, larger resolutions and higher frame-rates result in video content that contains much higher spatial frequency content, which motivates the use of more advanced motion models. While this paper is focusing on TFI, we believe that video compression and interpolation can be seamlessly integrated in the future.

Partly due to their low computational complexity and straightforward implementation, most existing TFI methods employ block-matching algorithms; we refer to [2] for an extensive review. Inherent block-artefacts in the interpolated frames are mitigated using so-called overlapped block motion compensation (OBMC) [3]. The downside of OBMC is that it unnecessarily lowers the high-frequency content of the final interpolated image. To further improve the quality of the upsampled frames, best-performing TFI schemes employ time-consuming texture optimization steps on the interpolated frame [4]. Veselov and Gilmutitdinov [5] propose a bilateral motion estimation scheme which finds the best motion vector for each block in the target frame grid; there is no occlusion-handling, and frame-rates can only be doubled. In higher resolution content, the size of disoccluded regions becomes larger, and hence an appropriate occlusion handling becomes more critical. Lu *et al.* [6] propose a *block-based* TFI method where they identify occluded regions, and propose a weighted OBMC to interpolate the upsampled frame.

Recent advances in optical flow estimation both in terms of quality [7], as well as reduction in computational complexity [8], motivate employing optical flow fields for frame interpolation. Herbst *et al.* [9] compute both a forward and backward optical flow between the two reference frames (*reference-anchored* motion), and then bidirectionally interpolate the target frame by independently mapping texture information from the two reference frames. Raket *et al.* [10] propose a symmetric total-variation (TV-L^1) optical flow at the target frame. While this *target-anchored* motion only requires the estimation of one flow field, it is not possible to explicitly handle disoccluded regions.

All reference-anchored TFI methods with *bidirectional* prediction we are aware of require the computation of a forward and a backward flow between the two reference frames. In contrast, the proposed TFI scheme only requires the forward flow, and the backward flow is constructed through an operation we call *motion inference*; this process not only halves the computation time for motion estimation, but also has an impact on the geometrical consistency of the bidirectional prediction process, as discussed in [11].

The proposed method inherits key properties from our earlier work on TFI [12]; in particular, we use information about motion discontinuities to handle regions around moving objects (i.e., disocclusions and double mappings). The two main contributions of the present work are:

- *Triangular mesh sparsification*, which adapts the triangle size of the affine mesh based on motion complexity (Sect. III); this reduces the computational complexity (see Sect. VII-A) of the triangle mapping process, which is the most time-consuming part of the proposed method;
- We create a *background motion layer* in regions that get disoccluded between the reference frames (Sect. V-A), which allows for temporally consistent interpolation;

To the best of our knowledge, the proposed *base-anchored mesh* BAM-TFI scheme is the first to address the problem of *temporal consistency* of an arbitrary number of interpolated frames in traditionally problematic regions around moving objects.

2016 IEEE 18th International Workshop on Multimedia Signal Processing (MMSP), Sept.21–23, 2016, Montreal, Canada. 978-1-4799-5896-2/14/$31.00

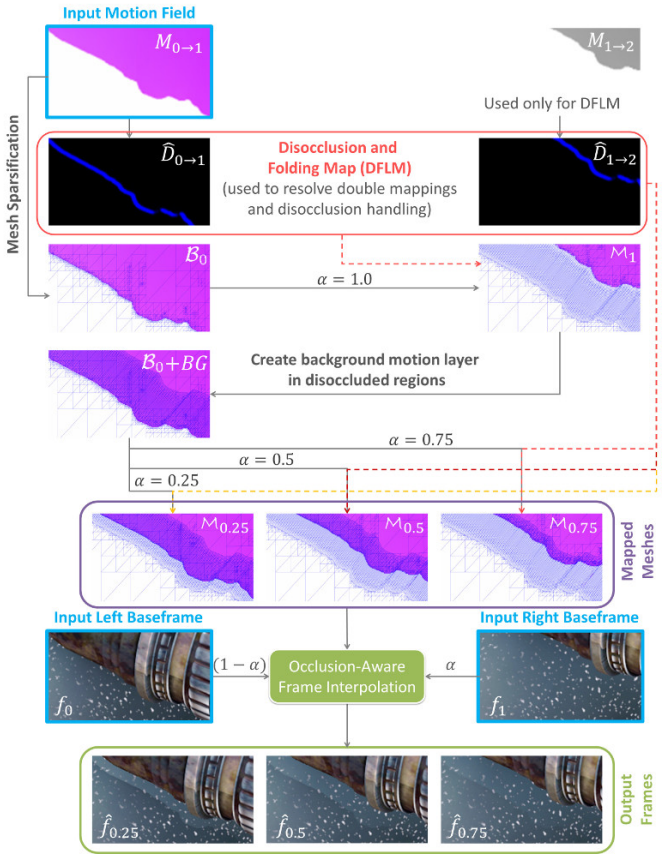


Fig. 1. Overview of the proposed base-anchored mesh (BAM) TFI method. We refer to the text for a description of the different parts.

II. Method Overview and Notations

In this section, we give an overview over the proposed *base-anchored motion* (BAM-TFI) method, and introduce notation used throughout the paper; we use Fig. 1 to guide the overview. Inputs to the method are two *base frames* f_0 and f_1, and the (estimated) motion field between them, $M_{0\to1}$; the notation $M_{i\to j}$ is used throughout the paper to denote a motion field anchored at frame f_i, pointing to frame f_j. From the two reference frames and the motion field connecting them, the aim is to interpolate frames f_α at time instance $\alpha \in [0,1]$; for example, the standard case of doubling the framerate that is considered in the literature is obtained by setting $\alpha = 0.5$. We highlight that the proposed framework allows for arbitrary upsampling factors.

The quality of the proposed method depends on the quality of the input motion fields. In particular, the motion fields needs to have sharp discontinuities around moving object boundaries; this is because we use motion discontinuity information to reason about foreground objects in order to resolve double mappings and disoccluded regions. For this purpose, we estimate a *disocclusion and folding map* (DFLM) on input motion fields $M_{i\to j}$, denoted as $\hat{D}_{i\to j}$, which is based on the divergence of the motion field.

We start by applying a triangular mesh sparsification algorithm (see Sect. III), which partitions the base motion field $M_{0\to1}$ into a *base* mesh $\mathcal{B}_0$. This base mesh holds a collection of K vertices $\{V_0^k\}, k \in \{0, K-1\}$, each of which holds motion vectors $\boldsymbol{u}_{0\to\alpha}(V_0^k)$ "linking" f_0 with *any* f_α we wish to interpolate (under constant motion assumption); in addition, it holds motion vectors for the special case of $\alpha = 1$, which links f_0 with f_1. These vertices are connected to form triangles, whose affine motion approximates the underlying motion field. While around moving object boundaries, even a triangle of size 1×1 is not able to describe the motion because of the discontinuity, one can expect that the affine motion from relatively large triangles is able to well approximate the "correct" motion within objects.

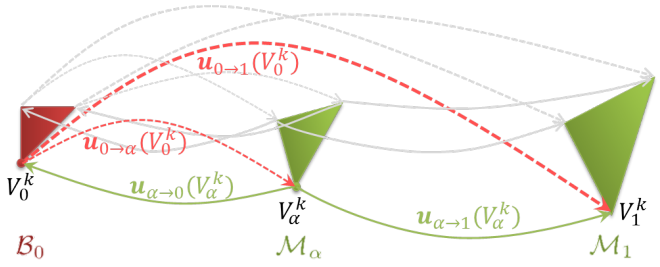
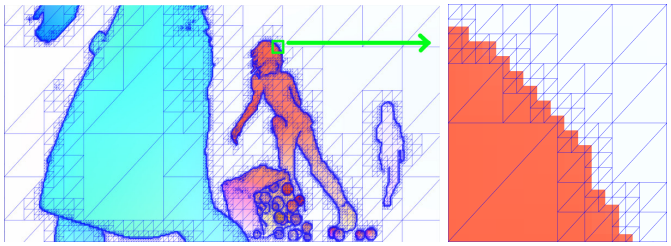


Fig. 2. Illustration how triangles of the base mesh and mapped meshes are linked via motion vectors. Dashed red arrows show base anchored motion, whereas green solid lines show mapped mesh motion for $\mathcal{M}_\alpha$, which links each vertex of the mesh with the preceding and succeeding reference frame.

In the proposed scheme, we also use the concept of a *mapped* mesh $\mathcal{M}_\alpha$, which is obtained by mapping the base mesh $\mathcal{B}_0$ to frame f_α. The main difference between a mapped mesh and the base mesh is the motion vectors its vertices hold. The mapped mesh contains vertices whose motion vectors $\boldsymbol{u}_{\alpha\to0}(V_\alpha^k)$ and $\boldsymbol{u}_{\alpha\to1}(V_\alpha^k)$ link it with *both* the preceding and the succeeding reference frames f_0 and f_1, respectively; this is illustrated in Fig. 2. Sect. IV is concerned with the fundamental operation of obtaining mapped motion vectors. Setting $\alpha = 1$, we map the base mesh from f_0 to f_1; the importance of the obtained $\mathcal{M}_1$ is that it reveals regions that get disoccluded between the two reference frames. Those are the regions for which no motion vector exists, and need special care in order to obtain a temporally consistent motion assignment. For such regions, we produce a "background" motion layer by adding new vertices to the base mesh that follow background motion. The purpose of this additional background layer is that if mapped to intermediate frames f_α in between the two base frames, temporally consistent motion will be assigned in disoccluded regions for varying values of α; this is illustrated in the bottom row of Fig. 1.

In $\mathcal{M}_\alpha$, triangles might overlap in regions where a foreground object moves on top of a background object. We use the DFLM (second row of the figure) to resolve such double mappings. Using a triangle ID check to identify regions that are not visible in either of the reference frames, we are able to switch from bidirectional to unidirectional prediction of the interpolated target frame(s) f_α in regions that are only visible in one of the reference frames.



(a) Triangular Basemesh (b) Crop

Fig. 3. Example basemesh created by the proposed mesh sparsification algorithm, superimposed on the (color-coded) dense motion field. Around motion discontinuities, triangles are split up to 1×1, whereas they are larger in regions of smooth (affine) motion.

III. Triangular Mesh Sparsification

In our earlier works ([11], [12]), we employed a triangular mesh with a *fixed* triangle size of 1×1. In regions of smooth motion, one can expect that the triangle size can be increased without significantly degrading the quality of the motion field. We note that we employ an *indexed mesh* structure, which allows for an efficient GPU implementation in the future.

Besides its positive impact on computational complexity, sparsifying the motion field has other interesting benefits when it comes to compression. In the long term, we envisage the proposed TFI method being used in a (highly scalable) video compression system, for which sparsity translates to compressibility; we leave this interesting direction for future research.

Algorithm 1 shows the pseudocode of the proposed triangular mesh sparsification algorithm. We start by partitioning $M_{0\to1}$ into cells of size L, where L is the largest allowed cell size. Then, each cell is split up into two triangles $T_{i,j,p,L}$, where (j,i) denote the upper left coordinates of the cell, and $p=0$ or $p=1$ are used to distinguish between the upper left and the lower right triangle of the cell. In other words, the coordinates of the three vertices of a triangle $T_{i,j,p,L}$ are:

$$\begin{aligned} V^0_{T_{i,j,p,L}}(x,y) &= (j+L,i) \\ V^1_{T_{i,j,p,L}}(x,y) &= (j,i+L) \\ V^2_{T_{i,j,p,L}}(x,y) &= \begin{cases} (j,i) & \text{if } p=0 \\ (j+L,i+L) & \text{if } p=1 \end{cases} \end{aligned} \tag{1}$$

Essentially, the proposed mesh sparsification algorithm starts with a largest cell size $L=2^N$, and splits the triangles of each cell up until they are "smooth". In this work, a triangle $T_{i,j,p,L}$ is considered *smooth* if for all (integer) locations $\mathbf{m}$ covered by the triangle, the interpolated motion $\boldsymbol{u}_{aff}[\mathbf{m}]$, obtained by *affine* interpolation of the motion of the three vertices of the considered triangle, predicts the original motion $\boldsymbol{u}[\mathbf{m}]$ with a prediction error lower than $thresh$; in the proposed work, we set $thresh = \frac{1}{2}$. Every triangle gets assigned a *unique* identifier (TID), which will be useful for creating visibility masks (see Sect. VI-B).

Algorithm 1 Mesh Sparsification Algorithm

```
procedure CreateSparseTriangularMesh
    L ← 2^N                                   ▷ Start with largest cell length
    for i = 0 to height step L do
        for j = 0 to width step L do
            CreateSmoothTriangle(T_{i,j,0,L})
            CreateSmoothTriangle(T_{i,j,1,L})
        end for
    end for
end procedure

function CreateSmoothTriangle(T_{i,j,p,L})
    if ||u[m] − u_aff[m]||_2 < thresh ∀m ∈ T_{i,j,0,L} then
        Create triangle T_{i,j,p,L}, assign unique TID
    else
        L ← L/2                               ▷ Assign new cell length
        if p == 0 then                        ▷ Split upper left triangle
            CreateSmoothTriangle(T_{i,j,0,L})
            CreateSmoothTriangle(T_{i,j,1,L})
            CreateSmoothTriangle(T_{i,j+L,0,L})
            CreateSmoothTriangle(T_{i+L,j,0,L})
        else                                  ▷ Split lower right triangle
            CreateSmoothTriangle(T_{i+L,j,1,L})
            CreateSmoothTriangle(T_{i,j+L,1,L})
            CreateSmoothTriangle(T_{i+L,j+L,1,L})
            CreateSmoothTriangle(T_{i+L,j+L,0,L})
        end if
    end if
end function
```

IV. Mesh Mapping: Inversion and Inference

We now describe how from the base mesh $\mathcal{B}_0$, we construct a mapped mesh $\mathcal{M}_\alpha$ at *any* time instance α in between the two reference frames. In particular, we present how mesh *inversion* and *inference* enable us to obtain motion relating each vertex of the mapped mesh with corresponding locations in the preceding and the succeeding reference frame (see Fig. 2).

Let $\boldsymbol{u}_{i\to j}(V^k_i)$ denote the motion vector of vertex k at time instance i, pointing to the corresponding location of the vertex V^k_j at time instance j.

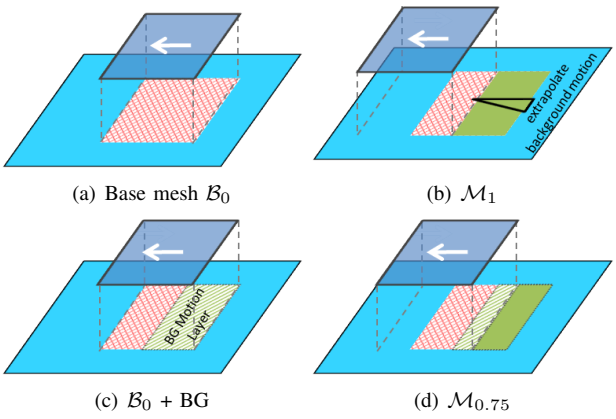
Then, the motion relating *any* vertex V^k_α of the mapped mesh $\mathcal{M}_\alpha$ to its location in the preceding reference frame f_0 (V^k_0) is obtained via *scaling* (by α) and *inversion* of the base mesh motion vectors:

$$\begin{aligned} \boldsymbol{u}_{\alpha\to0}(V^k_\alpha) &= -\boldsymbol{u}_{0\to\alpha}(V^k_0) \\ &= -\alpha\boldsymbol{u}_{0\to1}(V^k_0). \end{aligned} \tag{2}$$

The motion relating V^k_α to the corresponding location in the succeeding reference frame f_1 (V^k_1), is obtained performing an operation we call *mesh inference*:

$$\begin{aligned} \boldsymbol{u}_{\alpha\to1}(V^k_\alpha) &= \boldsymbol{u}_{\alpha\to0}(V^k_\alpha) + \boldsymbol{u}_{0\to1}(V^k_0) \\ &= -\alpha\boldsymbol{u}_{0\to1}(V^k_0) + \boldsymbol{u}_{0\to1}(V^k_0) \\ &= (1-\alpha)\boldsymbol{u}_{0\to1}(V^k_0). \end{aligned} \tag{3}$$

(3) exposes how motion vectors $\boldsymbol{u}_{\alpha\to1}$ are obtained via *composition* of scaled and inverted $\boldsymbol{u}_{\alpha\to0}$ (2) and the "parent" motion $\boldsymbol{u}_{0\to1}$; this intimately links the forward and backward



(a) Base mesh $\mathcal{B}_0$ (b) $\mathcal{M}_1$ (c) $\mathcal{B}_0$ + BG (d) $\mathcal{M}_{0.75}$

Fig. 4. Illustration of the disocclusion region motion backpropagation algorithm; we refer to the text for details.

pointing motion fields and guarantees geometrical consistency of the bidirectional prediction process.

V. Temporal Consistency around Moving Objects

In the previous section, we have shown how the base mesh can be mapped to any target frame f_α. In this mapped mesh $\mathcal{M}_\alpha$, on the *leading* side of moving objects, there will be regions where foreground triangles overlap with background triangles. Furthermore, in regions on the *trailing* side of objects in motion (disocclusion), the affine interpolated motion between foreground and background vertices is non-physical. In Sect. V-A, we show how to assign more realistic motion in disoccluded regions by creating a background motion layer; in Sect. V-B, we explain how the foreground triangle can be identified in regions where multiple triangles overlap.

A. Creating a Background Motion Layer in Disoccluded Regions

One of the most difficult aspects for ensuring temporal consistency is in regions that get disoccluded between the two reference frames. Most existing TFI methods use heuristics to fill in such regions, which can lead to visually disturbing artifacts when the upsampled video is viewed. The higher the frame upsampling factor, the more critical a temporally consistent interpolation in disoccluded regions becomes, since inconsistent motion can lead to visually disturbing artefacts. With the aid of Fig. 4, we outline the proposed procedure of obtaining temporally consistent motion in disoccluded regions.

Under the constant-velocity assumption that any TFI method using two reference frames is making, the area of disocclusion is monotonically increasing as objects transition from the preceding reference frame f_0 to the succeeding reference frame f_1. This implies that mapping the motion from f_0 to f_1 exposes all regions that get disoccluded, as shown in Fig. 4b. We want to find the object boundary in the succeeding reference frame, which outlines the end of the disoccluded region. We split up triangles at the motion discontinuity in f_1 by creating new vertices. We then assign extrapolated background motion to such "split" vertices. If mapped back to the base mesh $\mathcal{B}_0$, they create a background motion layer; this is illustrated in Fig. 4c, where in addition to the foreground motion, there is an *additional* background motion layer (dashed green lines). The motion of this layer will be applied in all regions of the target frame f_α that otherwise have no motion assigned (solid green region in Fig. 4d).

B. Identifying Local Foreground Triangles

In the mapped mesh $\mathcal{M}_\alpha$, triangles belonging to the foreground object might overlap with triangles of the background object; this happens on the leading side of moving objects. We resolve such double mappings using the same insight as used in our earlier work [12]. In essence, the motion of the triangle in the preceding reference frame which maps the motion discontinuity from $\hat{D}_{0\to1}$ closer to $\hat{D}_{1\to2}$ is identified as the foreground triangle; this is because motion discontinuities "travel" with the foreground object. The main difference is that whereas in our earlier work, double mappings were resolved on a per-pixel basis, here we resolve them on a per-triangle basis, which is particularly useful in regions of smooth motion.

We store this important information about foreground triangle IDs in a *foreground TID map*, denoted as F_α; any integer location $\mathbf{m}$ that is covered by the identified foreground triangle gets assigned the corresponding triangle ID.

VI. Weighted, Occlusion-Aware Frame Interpolation

We now explain how the mapped mesh $\mathcal{M}_\alpha$ is used to perform an occlusion-aware, bidirectional interpolation of the target frame f_α.

A. From Mapped Mesh to Prediction Motion Fields

Once all double mappings in $\mathcal{M}_\alpha$ are resolved, we compute the backward and forward prediction motion fields $\hat{M}_{\alpha\to0}$ and $\hat{M}_{\alpha\to1}$. This is done by assigning each integer location $\mathbf{m}$ the affine interpolated motion ($\boldsymbol{u}^{aff}_{\alpha\to k}$) of the foreground triangle:

$$\hat{M}_{\alpha\to k}[\mathbf{m}] = \boldsymbol{u}^{aff}_{\alpha\to k}(\mathbf{m}). \tag{4}$$

B. Visibility Mask

We use triangle IDs (TID) to assess whether a particular location $\mathbf{m}$ is hidden in either of the reference frames. This is done by a simple check whether the identified foreground TID at location $\mathbf{m}$ in f_α (i.e., $F_\alpha[\mathbf{m}]$) is the same as the one if $\mathbf{m}$ is mapped to f_0 or f_1 using $\hat{M}_{\alpha\to k}[\mathbf{m}]$. We compute a *visibility mask* $I^k_\alpha[\mathbf{m}]$ which stores this important information:

$$I^k_\alpha[\mathbf{m}] = \begin{cases} 1 & F_\alpha[\mathbf{m}] = F_k\left[\mathbf{m} + \hat{M}_{\alpha\to k}[\mathbf{m}]\right] \\ 0 & \text{otherwise} \end{cases}, \tag{5}$$

C. Motion-Compensated Temporal Frame Interpolation

We use a weighted bidirectional prediction of the motion compensated reference frames, denoted as $f_{0\to\alpha}$ and $f_{1\to\alpha}$, whenever the location $\mathbf{m}$ is visible in either *both* or *neither* of the reference frames, and switch to uni-directional prediction whenever a location is only visible in one reference frame.

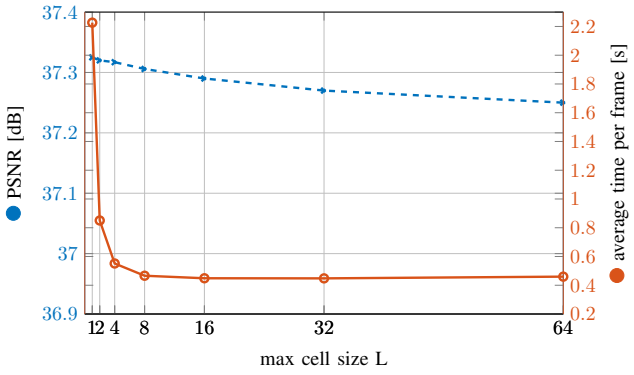


Fig. 5. Comparison between average per-frame processing time (orange solid line, in secs) and reconstructed PSNR (blue dashed line) as a function of maximum allowed cell size L.

$$\hat{f}_\alpha[\mathbf{m}] = \begin{cases} (1-\alpha) f_{0\to\alpha}[\mathbf{m}] + \alpha f_{1\to\alpha}[\mathbf{m}] & I^0_\alpha[\mathbf{m}] = I^1_\alpha[\mathbf{m}] \\ f_{k\to\alpha}[\mathbf{m}] & \text{otherwise} \end{cases}, \tag{6}$$

where k refers to the reference frame where the location $\mathbf{m}$ is visible.

VII. Experimental Results and Discussion

In this section, we evaluate the performance of the proposed BAM-TFI scheme. We start by assessing the impact of the proposed mesh sparsification algorithm. Next, we compare the TFI performance in terms of quality and computational complexity of the proposed scheme with a variety of other state-of-the-art TFI schemes. Lastly, we show qualitative results of the temporal consistency of the proposed TFI method.

A. Mesh Sparsification: Impact of Maximum Triangle Size

The proposed mesh sparsification allows to trade off computational complexity and memory requirements with image reconstruction quality. For evaluation, we use the data set used in [12], which contains a variety of common natural test sequences with different types of motion. We selected 12 sets of 21 frames each, and dropped all the odd indexed ones; this results in a total of 120 frames where a ground truth frame exists. For each even frame pair, we estimate motion using MDP-flow [7]. We choose a frame upsampling factor of 8, which interpolates 7 frames in between the two reference frames. For each frame pair, we compute the PSNR of the center frame ($\alpha = 0.5$), where a ground truth frame exists.

Fig. 5 shows the impact of maximum cell size L (between 1 and 64) on processing time (orange circles) and reconstruction quality (blue crosses). One can observe that the maximum allowed cell size L has very little impact on the reconstruction quality; this is due to the fact that around moving objects, where the motion is expected to be less smooth, the triangles are small irrespective of the L. Perhaps more interesting is the fact that in terms of processing times, the "sweet spot" on the present test sequence is for a cell size of $L = 32$, and then increases slightly for larger cell sizes. This is because most of the cells are broken up to 32×32 anyways, but the recursive splitting algorithm has more passes for larger maximum cell sizes, which takes (unnecessary) time. It can be expected that for larger resolutions, the sweet spot would shift to larger maximum cell sizes.

TABLE I
Average per-frame processing time (in sec) on all the frames from the test set, split up in motion estimation (ME) and Frame Interpolation (FI), as well as the average PSNR obtained on 120 interpolated frames.

Method	CPU	RAM	PSNR	ME	FI	Total
Jeong [4]	2.8GHz	8GB	35.1	410.2	498.9	909.1
Veselov [5]	2.6GHz	8GB	35.7	32.4	2.1	34.5
Lu [6]	2.4GHz	6GB	35.1	96.2	18.1	114.3
BOA-TFI [12]†	3.2GHz	8GB	37.1	355.4†	8.2	363.6
BAM-TFI†	3.2GHz	8GB	**37.3**	355.4†	**0.5**	355.9
BAM-TFI⋆			36.5	**7.0**⋆	**0.5**	**7.5**

† Motion fields estimated using MDP [7].
⋆ Motion fields estimated using EPIC flow [8].

B. Comparison with other TFI schemes

To highlight the high interpolation quality of the proposed scheme, we provide results for other state-of-the-art TFI schemes ([4], [5], [6]), as well as our earlier TFI work [12]. Table I shows the average computation time and average PSNR obtained by the tested TFI schemes on the same test set as used in the previous section.

We note that all the methods we compare ourselves to had as a design objective to create high quality interpolated frames. One can see that the proposed scheme outperforms current state-of-the-art both in terms of quality and processing time. We further ran our method using EPIC flow [8], which is around 50 times faster than MDP-flow [7]; yet, the proposed method still outperforms other state-of-the-art, while having the lowest processing times.

C. Temporal Consistency

In this section, we assess the temporal consistency of the proposed TFI framework, which is particularly important around moving objects. We also show the effectiveness of the proposed disocclusion region backprojection algorithm, whose goal is to assign temporally consistent motion in regions of disocclusion. In order to reduce the impact of the (quality of) the motion field estimator, we use challenging sequences from the Sintel sequence [13], where the ground truth motion is known. Fig. 6 contains crops of various sequences from the Sintel dataset, for a frame upsampling factor of 4.

Out of the methods we compare ourselves to, only BOA-TFI [12] can readily accommodate frame upsampling factors larger than 2; this framework is already tailored to create consistent results, and is probably above average performance around moving objects. Nonetheless, the BAM-TFI framework we propose in this paper creates more consistent results around moving objects, especially around thin moving objects.

(a) Input f_{16} and f_{17}, and $\frac{1}{2}(f_{16}+f_{17})$ (b) $\hat{f}_{16.25}$, $\hat{f}_{16.5}$, and $\hat{f}_{16.75}$ using BAM-TFI (c) $\hat{f}_{16.25}$, $\hat{f}_{16.5}$, and $\hat{f}_{16.75}$ using BOA-TFI

(d) Input f_6 and f_7, and $\frac{1}{2}(f_6+f_7)$ (e) $\hat{f}_{6.25}$, $\hat{f}_{6.5}$, and $\hat{f}_{6.75}$ using BAM-TFI (f) $\hat{f}_{6.25}$, $\hat{f}_{6.5}$, and $\hat{f}_{6.75}$ using BOA-TFI

(g) Input f_{25} and f_{26}, and $\frac{1}{2}(f_{25}+f_{26})$ (h) $\hat{f}_{25.25}$, $\hat{f}_{25.5}$, and $\hat{f}_{25.75}$ using BAM-TFI (i) $\hat{f}_{25.25}$, $\hat{f}_{25.5}$, and $\hat{f}_{25.75}$ using BOA-TFI

Fig. 6. Crops of various sequences from the Sintel dataset [13] (from top to bottom market_2, ambush_6, and bamboo_2). Each row first shows the two input frames, as well as the average of the two input frames, to give an idea of the motion. The next three frames are the interpolated frames produced by the proposed BAM-TFI method, followed by the results obtained using BOA-TFI [12].

VIII. Conclusions and Future Work

This paper introduces the concept of base-anchored motion using a sparsified mesh for temporal frame interpolation, which we refer to as BAM-TFI. We propose a triangular mesh sparsification algorithm, which is able to significantly reduce the computational complexity of the proposed TFI method with almost no degradation in reconstruction quality. The second contribution is a method of creating a background motion layer in disoccluded regions, which guarantees temporally consistent motion assignments in regions that get uncovered between the two reference frames. Compared to current state-of-the-art TFI methods, the proposed method has higher quality and lower processing times, and is able to create temporally consistent interpolated frames.

In future work, we plan to include higher motion model orders to account for motion that is not constant, which opens interesting links to highly scalable video compression. We further intend to implement the proposed indexed mesh structure into the GPU, which promises further reduction in processing time.

References

[1] C. N. Cordes and G. Haan, "Key requirements for high quality picture-rate conversion," *SID Symposium Digest of Technical Papers*, vol. 40, no. 1, pp. 850–853, 2009.

[2] S. Dikbas and Y. Altunbasak, "Novel true-motion estimation algorithm and its application to motion-compensated temporal frame interpolation," *IEEE Trans. Image Proc.*, vol. 22, no. 8, pp. 2931–2945, 2013.

[3] M. T. Orchard and G. J. Sullivan, "Overlapped block motion compensation: An estimation-theoretic approach," *IEEE Trans. Image Proc.*, vol. 3, no. 5, pp. 693–699, 1994.

[4] S.-G. Jeong, C. Lee, and C.-S. Kim, "Motion-compensated frame interpolation based on multihypothesis motion estimation and texture optimization," *IEEE Trans. Image Proc.*, pp. 4497–4509, 2013.

[5] A. Veselov and M. Gilmutdinov, "Iterative Hierarchical True Motion Estimation for Temporal Frame Interpolation," *IEEE Int. Workshop on Multimedia Sig. Proc.*, 2014.

[6] Q. Lu, N. Xu, and X. Fang, "Motion-Compensated Frame Interpolation With Multiframe Based Occlusion Handling," *Journal of Display Technology*, vol. 11, no. 4, 2015.

[7] L. Xu, J. Jia, and Y. Matsushita, "Motion detail preserving optical flow estimation," *IEEE Trans. Patt. Anal. and Mach. Intell.*, pp. 1744–1757, 2012.

[8] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Epicflow: Edge-preserving interpolation of correspondences for optical flow," *Proc. IEEE Conf. Comp. Vis. and Patt. Rec.*, 2015.

[9] E. Herbst, S. Seitz, and S. Baker, "Occlusion reasoning for temporal interpolation using optical flow," *Dept. of Comp. Science and Eng., University of Washington, Tech. Rep. UW-CSE-09-08-01*, 2009.

[10] L. L. Rakêt, L. Roholm, A. Bruhn, and J. Weickert, "Motion compensated frame interpolation with a symmetric optical flow constraint," *Advances in Visual Computing*, pp. 447–457, 2012.

[11] D. Rüfenacht, R. Mathew, and D. Taubman, "A Novel Motion Field Anchoring Paradigm for Highly Scalable Wavelet-based Video Coding," *IEEE Trans. Image Proc.*, 2015.

[12] ——, "Occlusion-Aware Temporal Frame Interpolation in a Highly Scalable Video Coding Setting," *APSIPA Trans. Signal and Information Proc.*, 2016.

[13] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," *European Conf. on Comp. Vis.*, pp. 611–625, Oct. 2012.