FRAME INTERPOLATION WITH OCCLUSION DETECTION USING A TIME COHERENT SEGMENTATION

Rida Sadek[1], Coloma Ballester[1], Luis Garrido[2], Enric Meinhardt[1] and Vicent Caselles[1]

[1]*Department of Technology, Pompeu-Fabra University, Roc Boronat, Barcelona, Spain*

[2]*Department of Applied Mathematics and Analysis, Barcelona University, Barcelona, Spain*

{*rida.sadek,coloma.ballester*}*@upf.edu,lluis.garrido@ub.edu,*{*enric.meinhardt,vicent.caselles*}*@upf.edu*

Keywords: Frame interpolation, optical flow, occlusion

Abstract: In this paper we propose an interpolation method to produce a sequence of plausible intermediate frames between two input images. The main feature of the proposed method is the handling of occlusions using a time coherent video segmentation into spatio-temporal regions. Occlusions and disocclusions are defined as points in a frame where a region ends or starts, respectively. Out of these points, forward and backward motion fields are used to interpolate the intermediate frames. After motion-based interpolation, there may still be some holes which are filled using a hole filling algorithm. We illustrate the proposed method with some experiments.

1 INTRODUCTION

Our purpose in this paper is to propose an interpolation method to produce a sequence of plausible intermediate frames between two input images with applications to slow camera motion for smooth playback of lower frame rate video, smooth view interpolation and animation of still images. Recent progress in optical flow estimation provides optical flows of sufficient quality for intermediate frame interpolation (Brox et al., 2004; Brox and Malik, 2010; Sun et al., 2010). One of the main difficulties that has to be tackled in image interpolation is the occlusion effects. Points visible at time t that occlude at time $t+1$ should not have a corresponding point at frame $t+1$. While points that appear at time $t+1$ should have no correspondent at time t. Image interpolation algorithms have to detect such occlusions in order to correctly decide how to interpolate.

Most current optical flow estimation methods do not directly compute occlusions and assign an optical flow to each pixel of each frame. If we blindly use the optical flow, artifacts are generated. They are specially visible at moving occlusion boundaries. This difficulty is usually handled by analyzing the forward and backwards motion fields in order to decide from which image we interpolate. In this paper we propose to address the occlusion effects by using a spatio-temporal segmentation of the video sequence. This segmentation allows us to interpret the sequence as a set of spatio-temporal regions whose temporal boundaries give us information about their birth and death. As a result, we are able to extract a set of candidate occluding and disoccluding points. Forward and backward motion fields are then used to interpolate the intermediate image taking into account the latter points. After motion-based interpolation, there may still be some holes which are filled using a suitable hole filling algorithm (Criminisi et al., 2004; Arias et al., 2011).

1.1 Previous work

There are many papers devoted to intermediate frame interpolation. Many of them are based on establishing correspondences between consecutive pairs of images. They are usually computed by means of block based motion estimation or dense optical flow. The former has been applied to frame-rate conversion for digital television. The image is divided into a set of non-overlapping blocks and forward and/or backward motion compensation is performed to create the interpolated frame (Cafforio et al., 1990). Occlusion effects are handled by analyzing the forward and backwards motion fields in order to decide from which frame to interpolate. In (Dane and Nguyen, 2004) the authors conclude that the averaging technique is appropriate if forward and backward prediction errors are equal. In other cases, they conclude with theoretical results that the filter taps associated to the motion compensation have to be adapted to the reliability of the motion vectors.

Frame interpolation also has been tackled by means of dense optical flow. Most of current optical flow approaches compute the forward field based on the work of Horn and Schunk (see (Baker et al., 2011)). Other authors compute a symmetric optical flow by means of a penalty in their model (Alvarez et al., 2007; Ince and Konrad, 2008) which permits to define occlusion regions.

In the context of frame interpolation, (Baker et al., 2011) propose a simple method for frame interpolation and occlusion handling based on forward *splatting* which only uses the forward flow. In (Herbst et al., 2009) the authors enhance the latter approach by means of forward and backward splatting. Occlusion is detected by assessing the flow consistency of the forward and backward flows at the intermediate frame. Holes are treated by extending, at the intermediate frame, the motion vectors of neighboring pixels using a Markov Random Field (MRF). (Linz et al., 2010) present a long-range correspondence estimation technique that includes SIFT, edge and symmetry data terms. Occlusion also is detected by assessing the coherence between the two flows. The interpolated image is created by using a graph-cut based approach that decides at each pixel from which image the color information is taken.

In (Mahajan et al., 2009) the authors present a method based on graph cuts which is based on the idea that a given pixel traces out a path in the source images. Their method can we viewed as an inverse optical flow algorithm: they compute where in the input images a given intermediate pixel comes from. Occlusion effects are again tackled by assessing the consistency between the forward and backward flows. The advantage of their method is that no holes are created and thus they need no treatment for them.

1.2 Overview of our algorithm

We briefly summarize the main Steps of our algorithm. Let $I(x,y,t)$ be a video sequence. For simplicity we consider that the video is sampled at times $t = 0,1,2,\ldots$. The image domain Ω is a rectangular grid in $\mathbb{Z}^2$.

Step 1 : We compute the forward (u,v) and backward (u^b,v^b) optical flows. For that we use the algorithm (Brox and Malik, 2010), see section 2.1. As an alternative, we can also use (Sun et al., 2010). Both estimate a forward flow between images t and $t+1$ and have no occlusion treatment. The binary file and the code, respectively, of both algorithms is publicly available.

Step 2 : Using the forward flow we compute a time consistent segmentation of the video sequence (see section 2.2).

Step 3 : Potential occlusions are then derived from Step 2. We then interpolate the intermediate frames using the idea of forward and backward splatting described in (Herbst et al., 2009), see section 2.3. The result may contain holes made of points which could not be interpolated.

Step 4 : The holes are filled-in using an inpainting strategy, see section 2.4.

2 ALGORITHM DESCRIPTION

2.1 Optical Flow Computation

For the computation of the optical flow we can use any optical flow producing good results. Our experiments have been done with the optical flow algorithm proposed in (Brox and Malik, 2010) which uses HOG (Histogram of Oriented Gradients) descriptors in order to be able to follow fast motions. Estimated motion vectors are required to follow descriptor matchings. As an alternative, we can also use (Sun et al., 2010).

2.2 Time Coherent Video Segmentation

As stated previously, the objective is to create a time consistent segmentation of the video sequence. We consider the video sequence as a volume of 3D data. Our segmentation is based on the simplified Mumford-Shah model. Given the video sequence $I(x,y,t)$, the simplified Mumford-Shah model approximates I by a piecewise constant function $\tilde{I}(x,y,t) = \sum_{O\in\mathcal{P}} m_O \chi_O(x,y,t)$ that minimizes the energy

$$\sum_{t=0}^{N}\sum_{x\in\Omega}(I(x,y,t)-\tilde{I}(x,y,t))^2+\lambda\sum_{O}\mathrm{Area}(\partial O), \quad (1)$$

where $\lambda > 0$, $\mathcal{P}$ is a partition of $\Omega\times\{0,\ldots,N\}$ into connected regions O, and *Area* corresponds to the area of the interface that separates two spatio-temporal regions. Note that the area for each interface is counted twice in (1), but this amounts only to a replacement of the value of the parameter λ by $\lambda/2$. As explained in (Koepfler et al., 1994), the parameter λ controls the number of regions of the segmentation. For a given partition $\mathcal{P}$, the constant m_O is the average of $I(x,y,t)$ in the region O.

We replace the classical notion of connectivity by a time compensated one. For that we construct a graph whose nodes are the pixels of the video, i.e. $\{(x,y,t) : (x,y) \in \Omega, t \in \{0,\ldots,N\}\}$. There are two types of edges in the graph: spatial and temporal ones. Spatial edges connect a pixel (x,y,t) to its 8-neighborhood in frame t. Temporal edges are defined using the pre-computed forward optical flow. If the flow vector for pixel (x,y,t) is (u,v), then we add to the graph an edge joining pixel (x,y,t) to pixel $(x',y',t') = (x+[u], y+[v], t+1)$, where the square brackets denote the nearest integer. This graph gives us the 3D neighborhood of each point (x,y,t). This permits an easy adaptation of the algorithm in (Koepfler et al., 1994)

For a given λ and following (Koepfler et al., 1994), the energy is optimized with a region merging strategy that computes a 2-normal segmentation. A 2-normal segmentation is defined by the property that merging any pair of its regions increases the energy of the segmentation. Notice that 2-normal segmentations are typically not local minima of the functional; however, they are fast to compute and useful enough for our purposes. The region merging strategy consists in iteratively coarsening a given pre-segmentation, which is stored as a region-adjacency graph. Each edge of this graph is marked by the energy gain that would be obtained by merging the corresponding pair of regions into one. Then, at each step of the algorithm the optimal merge – the one that leads to the best improvement of the energy – is performed, thus reducing the region adjacency graph by one region and one or more edges. The energy gain is recomputed for the neighbouring regions and the algorithm continues to merge as long as they produce some improvement of the energy functional. Note that the parameter λ of the energy functional controls the number of regions of the resulting segmentation. When finding 2-normal segmentations by region merging, this parameter can be automatically set by specifying directly the desired number of regions.

In practice, we do not know which value of λ will produce a good segmentation. For that reason we proceed as follows: we create a set of partitions that are obtained by successively increasing λ (e.g. dyadically). Each partition is computed by taking as input the previously obtained partition and merging regions as described in the previous paragraph. The algorithm starts with a low value of λ using the time-connected graph described above, and stops when the trivial partition is obtained. The history of all the mergings is stored in a (binary) tree, whose nodes represent each region of the segmentation at some iteration. The leafs of this tree are the pixels of the input video. The internal nodes of this tree are regions appearing at some iteration, and the root of the tree is the whole video. While the tree is being built, each node is marked with the value of λ at which the corresponding region has been created.

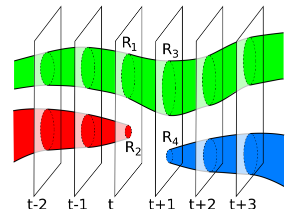

Figure 1: This figure illustrates the concept of tubes as described in the paper. We see a tube that ends at t, a tube that starts at $t+1$ and a tube that continues through frame t.

Once this tree is built, it can be cut at any desired value of λ in real-time, to produce segmentations at different scales. We call *tubes* the spatio-temporal regions of the resulting partition (see Figure 1). The tubes encode a temporally coherent segmentation of the objects in the video, which can be used for several purposes (e.g. tracking). We use them here in order to determine potential occlusions by analyzing their temporal boundaries. Any connected tube O has a starting and and ending times, denoted by T_O^s and T_O^e, respectively. The section of O at time t is denoted by $O(t) = \{(x,y) \in \Omega : (x,y,t) \in O\}$. Thus O starts (resp. ends) with the spatial region $O(T_O^s)$ (resp. $O(T_O^e)$).

2.3 Intermediate Frame Interpolation

Given two video frames I_0 and I_1 at times $t=0$ and $t=1$ respectively, our purpose is to interpolate an intermediate frame at time $t = \delta \in (0,1)$. For that issue the forward and backward motion fields are first estimated. In order to tackle with the occlusion effects we use the information of the temporal boundaries obtained with our time coherent segmentation algorithm. Intuitively, a given pixel (x,y,t) at $t=0$ is forward projected if it is not in a dying statio-temporal region. Similarly, a given pixel (x,y,t) at $t=1$ is backward projected if it is not in a spatio-temporal region that has birth there. Let us now go into the details of the algorithm.

The frame interpolation algorithm starts by marking all pixels to be interpolated as holes. Then two stages are performed: in the first stage a forward projection is done; in the second a backward projection is performed to (partially) fill in the holes that the first stage may have left.

For the first stage we use the forward optical flow (u,v) from $t=0$ to $t=1$. Let $\mathcal{F}(t=0)$ be the set of pixels of frame $t=0$ which do not belong to tubes that end at time $t=0$. This information is contained in the data structure that we developed for the spatio-temporal segmentation.

Let us describe the forward projection step. It is based on the idea of splatting described in (Herbst et al., 2009).

For a given pixel $(x,y,t=0)$, let $p_\delta = (x,y) + \delta(u,v)$ be the forward projected point of the pixel. Note that p_δ may be a non-integer point. Let $p_\delta^{00} = \lfloor p_\delta \rfloor$ be the pixel whose coordinates are given by the integer parts of the coordinates of p_δ, and let $p_\delta^{ab} = p_\delta^{00} + (a,b)$ for $(a,b) = (0,1),(1,0),(1,1)$, the four points bounding the square containing p_δ. We assign the flow (u,v) to each pixel p_δ^{ab} and compute its forward and backward projections: let them be $p_0^{ab} = p_\delta^{ab} - \delta(u,v)$ and $p_1^{ab} = p_\delta^{ab} + (1-\delta)(u,v)$. The values $I_0(p_0^{ab})$ and $I_1(p_1^{ab})$ are computed using bilinear interpolation.

The pixel $(x,y,t=0)$ is forward interpolated if pixel (x,y,t) belongs to $\mathcal{F}(t=0)$ or if

$$|I_0(p_0^{ab}) - I_1(p_1^{ab})| \leq \tau, \qquad (2)$$

where $\tau > 0$ is a pre-specified tolerance value (in our experiments, we take the value $\tau = 10$). If the previous conditions are not hold, then we do not forward interpolate.

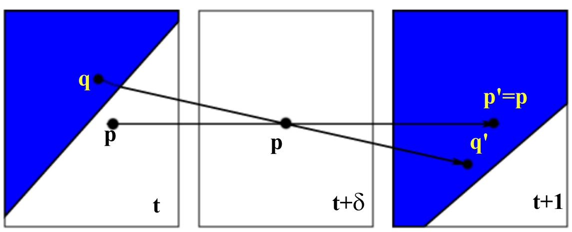

Figure 2: A case of inconsistent interpolation as explained in the text.

Equation (2) allows to deal with pixels which are part of objects that are visible in both frames I_0, I_1 (and thus belong to a continuing tube) but are occluded in frame I_1, inheriting the optical flow of the visible object as an effect of the regularization terms (see the pixel p in Figure 2). In that case, the correspondents of both points (p,t) and (q,t) in Figure 2 go through $(p, t+\delta)$ and we expect that the candidate interpolation from $I(p,t)$ and $I(p', t+1)$ is discarded because inconsistent gray level values, while the interpolation between $I(q,t)$ and $I(q', t+1)$ is retained because it satisfies (2).

In case the pixel $(x,y,t=0)$ can be forward interpolated the pixels at p_δ^{ab} are computed as

$$\tilde{I}_\delta(p_\delta^{ab}) = (1-\delta) I_0(p_0^{ab}) + \delta I_1(P_1^{ab}) \qquad (3)$$

and marked as interpolated.

The previous process is repeated for all pixels. We scan all pixels at $t=0$ from left to right and top to bottom. For each pixel we proceed as described above. In case multiple interpolated values are assigned to a intermediate pixel, we assign to it the average of all values $\tilde{I}_\delta(p_\delta^{ab})$ for $p_\delta^{ab} = (\overline{x}, \overline{y})$.

Once the first stage has been performed, the algorithm now uses the backward flow to deal with the holes the first step may have left. Note that all pixels labeled as interpolated (in the first stage) are not altered. The algorithm is basically the same as the one described before but in this case a backward projection is done, replacing $\mathcal{F}(t=0)$ by $\mathcal{F}(t=1)$. Here $\mathcal{F}(t=1)$ is defined as the set of tubes that do not begin at time $t=1$. This can be easily checked on the graph (which has been constructed using the forward flow).

2.4 Filling the Holes

Holes appear inevitably due to occlusions, disocclusions, or during the interpolation process due to the expansive or contractive character of the flow. Some occlusions may generate ending tubes at time t (which should not be images of the backward flow). Thus, we do not project those pixels forward for interpolation. Disocclusions are not images (and some of them may be starting tubes) of the forward flow and may also generate holes. To fill-in the remaining holes we use an inpainting strategy. Each pixel in a hole of I_δ is filled in by an exemplar based interpolation as in (Arias et al., 2011) (see also (Criminisi et al., 2004)). To fill a pixel in frame t we search for patches in the previous and next frames. To reduce the searching area we search in regions of frame t (resp. $t+1$) determined by the optical flows of pixels bounding the hole. The search of best patches can be accelerated using the Patch-Match algorithm (Barnes et al., 2009).

3 EXPERIMENTS

Let us display some experiments. Figure 3 shows the interpolation between two different frames of the sequence MiniCooper, available at http://vision.middlebury.edu/flow/data/. The person is closing the door at the back of the car. The maximum displacement is 17.28. From left to right and top to bottom, the first and last images belong to the original sequence. The four intermediate frames have been interpolated. In Figure 4 we show the holes generated by the interpolation process. The left image shows the holes after forward interpolation, the right image shows the remaining ones after backward interpolation. Those are the ones that are filled-in by inpainting. In Middlebury an intermediate frame is given for comparison. It corresponds to the fourth image in Figure 3. The RMSE is in this case 4.2440.

Figure 5 shows the interpolation between two different frames of the sequence Foreman. The person is opening the mouth and some regions are disoccluded. The maximum displacement is 9.8331. From left to right and top to bottom, the first and last images belong to the original sequence. The four intermediate frames have been interpolated. In Figure 6 we show the holes generated by the interpolation process. The left image shows the holes after forward interpolation, the right image shows the remaining ones after backward interpolation which we filled-in by inpainting.

Figure 7 shows the interpolation of four frames from a sequence where a placard is being disoccluded. Figure 8 shows the interpolation of an intermediate frame in the sequence Urban taken from Middlebury. The camera moves right and there are occlusions between buildings. The experiments displayed here can be found in http://www.dtic.upf.edu/∼cballester/demos/scm .

In our experiments the computation time per frame is around: 5 sec for optical flow computation, 0.55 sec for the intermediate frame interpolation and 1 min for the inpainting algorithm.

Figure 3: Intermediate frame interpolation. The first and last frames are original images. The intermediate ones are interpolated.

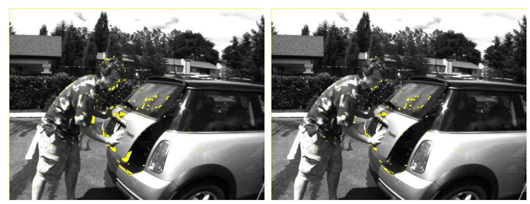

Figure 4: Left: The holes appearing in the first interpolated frame of Figure 3 after forward interpolation. Right: The remaining holes after backward interpolation.

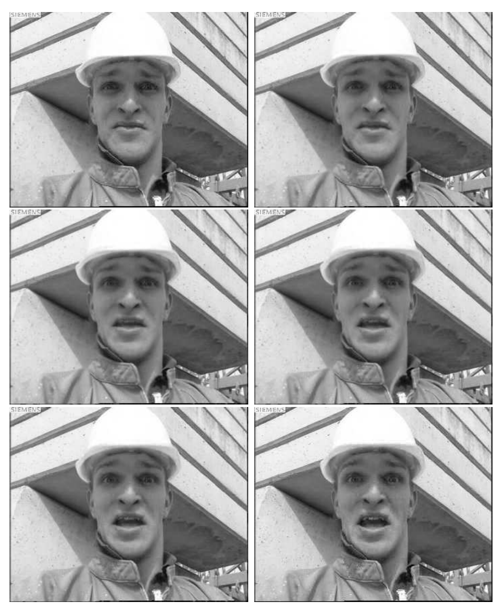

Figure 5: Intermediate frame interpolation. The first and last frames are original images. The intermediate ones are interpolated.

4 CONCLUSIONS

We have proposed an interpolation method to produce a sequence of plausible intermediate frames between two input images. The method is based on the exploitation of the optical flow and the handling of occlusions by means of a time coherent video segmentation which produces a set of spatio-temporal regions. Occlusions and dis-occlusions are identified as the temporal boundaries where a region ends or starts, respectively. With this we avoid the forward propagation of occluded regions. After forward propagation of continuing regions, holes appear and we use the backward flow to propagate information not belonging to starting regions from next frame. The few remaining holes are filled-in by inpainting. The method permits to avoid artifacts in occluding and disoccluding regions. The method is sensitive to the quality of the optical flow and requires flows of high quality.

ACKNOWLEDGEMENTS

We acknowledge partial support by MICINN project, reference MTM2009-08171, and by GRC reference

Figure 8: Intermediate frame interpolation. The first and third frames are original images. The middle one is interpolated.

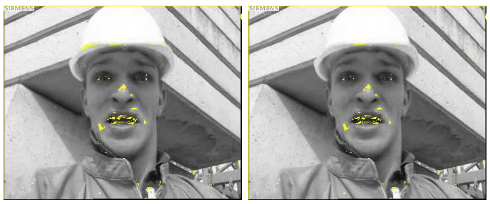

Figure 6: Left: The holes appearing in the last interpolated frame of Figure 3 after forward interpolation. Right: The remaining holes after backward interpolation.

Figure 7: Intermediate frame interpolation. The first and last frames are original images. The intermediate ones are interpolated.

2009 SGR 773. VC also acknowledges partial support by "ICREA Acadèmia" prize for excellence in research funded by the Generalitat de Catalunya.

REFERENCES

Alvarez, L., Deriche, R., Papadopoulo, T., and Sánchez, J. (2007). Symmetrical dense optical flow estimation with occlusions detection. *Int. Journal of Comp. Vis.*, 75:371–385.

Arias, P., Facciolo, G., Caselles, V., and Sapiro, G. (2011). A variational framework for exemplar-based image inpainting. *Int. Journal of Comp. Vis.*, 93:319–347.

Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M., and Szeliski, R. (2011). A database and evaluation methodology for optical flow. *Int. Journal of Comp. Vis.*, 92:1–31.

Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. B. (2009). PatchMatch: a randomized correspondence algorithm for structural image editing. In *Proc. of SIGGRAPH*, pages 1–11.

Brox, T., Bruhn, A., Papenberg, N., and Weickert, J. (2004). High accuracy optical flow estimation based on a theory for warping. In *ECCV*, pages 25–36.

Brox, T. and Malik, J. (2010). Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE Transactions on PAMI*, pages 500–513.

Cafforio, C., Rocca, F., and Tubaro, S. (1990). Motion compensated image interpolation. *IEEE Transactions on Communications*, 38:215–222.

Criminisi, A., Pérez, P., and Toyama, K. (2004). Region filling and object removal by exemplar-based image inpainting. *IEEE Trans. on IP*, 13:1200–1212.

Dane, G. and Nguyen, T. (2004). Motion vector processing for frame rate up conversion. In *IEEE ICASSP*, volume 3, pages 309–312.

Herbst, E., Seitz, S., and Baker, S. (2009). Occlusion reasoning for temporal interpolation using optical flow. In *Technical report UW-CSE-09-08-01*. Dept. of Comp. Sci. and Eng., University of Washington.

Ince, S. and Konrad, J. (2008). Occlusion-aware optical flow estimation. *IEEE Trans. on IP*, 17:1443–1451.

Koepfler, G., Lopez, C., and Morel, J. (1994). A multiscale algorithm for image segmentation by variational method. *SIAM Journal on Numerical Analysis*, 31:282–299.

Linz, C., Lipski, C., and Magnor, M. (2010). Multi-image interpolation based on graph-cuts and symmetric optical flow. In *15th International Workshop on Vision, Modeling and Visualization (VMV)*, pages 115–122.

Mahajan, D., Huang, F., Matusik, W., Ramamoorthi, R., and Belhumeur, P. (2009). Moving gradients: a path-based method for plausible image interpolation. In *ACM SIGGRAPH*, volume 28, pages 1–11.

Sun, D., Roth, S., and Black, M. (2010). Secrets of optical flow estimation and their principles. In *IEEE Conference on CVPR*, pages 2432–2439.