All at Once: Temporally Adaptive Multi-Frame Interpolation with Advanced Motion Modeling

Zhixiang Chi, Rasoul Mohammadi Nasiri, Zheng Liu
Juwei Lu, Jin Tang, Konstantinos N Plataniotis

In this document, we provide the proof and derivation of the cubic motion modeling equations in Section 1 and 7, additional analysis for motion relaxation in Section 2, architecture of the flow estimation network in Section 3, temporal consistency analysis in Section 4 and more visual examples in Section 6.

1 Proof of the cubic flow prediction

In this section, we provide the details in deriving equations of our proposed cubic motion model. We can consider pixel displacement representing an object motion in the real world. From physics of motion, we can formulate object displacement with a variable acceleration between two times of 0 and t as

$$s_t - s_0 = v_0 \times t + \frac{a_0}{2} \times t^2 + \frac{\Delta a_0}{6} \times t^3, \tag{1}$$

where v_0, a_0, and Δa_0 are the velocity, acceleration, and acceleration change rate estimated at time 0. s_t and s_0 are the positions for pixels at time 0 and t. We use this model for any time between 0 and 1 (for I_0 and I_1) to predict pixels displacement from I_0 to I_t for $0 < t < 1$. We are using I_0 as the reference point and Since $s_t - s_0$ represents the displacement between corresponding pixels at t and 0, it equals to the optical flow as $f_{0\to t}$. We first set t in (1) to obtain displacement to times 1 and -1 as

$$f_{0\to 1} = v_0 + \frac{a_0}{2} + \frac{\Delta a_0}{6}, \tag{2}$$

$$f_{0\to -1} = -v_0 + \frac{a_0}{2} - \frac{\Delta a_0}{6}, \tag{3}$$

From the above equations, we can compute a_0 by

$$a_0 = f_{0\to 1} + f_{0\to -1}. \tag{4}$$

We can follow the same approach to calculate a_1 by using I_1 as reference in (1) and substitute t as 0 and 2 to obtain

$$a_1 = f_{1\to 2} + f_{1\to 0}. \tag{5}$$

The acceleration change rate Δa_0 can be computed as $a_1 - a_0$; however this pixel-wise difference needs to be performed on corresponding pixels between two frames, not just on pixels in the same coordinate. To compute a_1 for pixels based on their location at I_0, we set t to 2 in (1) which computes $f_{0\to 2}$ as:

$$f_{0\to 2} = 2v_0 + 2a_0 + \frac{8 \times (a_1 - a_0)}{6}, \tag{6}$$

Using (2) and (6), we can calculate a_1 as

$$a_1 = f_{0\to 2} - 2 \times f_{0\to 1}, \tag{7}$$

which equals the equation (3) in the paper. Even though an extra step needed to compute $f_{0\to 2}$ our method is still efficient because of the one-shot structure.

To compute v_0 and show why the calculation in [1] does not hold for cubic motion modeling, we first rearrange (2) as:

$$v_0 = f_{0\to1} - \frac{a_0}{2} - \frac{a_1 - a_0}{6}. \tag{8}$$

(8) is the same as equation (4) in the paper, where a_0 and a_1 are both previously computed by the optical flows at reference point I_0. Then we substitute (4) to (8) as:

$$v_0 = f_{0\to1} - \frac{f_{0\to1} + f_{0\to-1}}{2} - \frac{a_1 - a_0}{6} \tag{9}$$

and rearrange:

$$v_0 = \frac{f_{0\to1} - f_{0\to-1}}{2} - \frac{a_1 - a_0}{6}. \tag{10}$$

The v_0 proposed in [1] has only the first term at the right of (10) and does not consider the acceleration change (the second term).

Finally, the optical flows from time 0 to a middle frame t_i (for $i \in [1..7]$ for 7 frame interpolation) can be computed by substituting v_0 in (1) with (10) as

$$f_{0\to t_i} = f_{0\to1} \times t_i + \frac{a_0}{2} \times (t_i^2 - t_i) + \frac{a_1 - a_0}{6} \times (t_i^3 - t_i). \tag{11}$$

and furthermore, a_0 and a_1 in the above equation can be replaced by (5) and (7) to write equation based on using only optical flows

$$f_{0\to t_i} = f_{0\to1} \times t_i + \frac{f_{0\to1} + f_{0\to-1}}{2} \times (t_i^2 - t_i) + \frac{f_{0\to2} - 2 \times f_{0\to1} - (f_{0\to1} + f_{0\to-1})}{6} \times (t_i^3 - t_i). \tag{12}$$

Similarly, optical flow from time stamp $t = 1$ to the middle frames t_i can be computed as the same manner as:

$$f_{1\to t_i} = f_{1\to0} \times \hat{t}_i + \frac{f_{1\to0} + f_{1\to2}}{2} \times (\hat{t}_i^2 - \hat{t}_i) + \frac{f_{1\to-1} - 2 \times f_{1\to0} - (f_{1\to0} + f_{1\to2})}{6} \times (\hat{t}_i^3 - \hat{t}_i), \tag{13}$$

where $\hat{t}_i = 1 - t_i$.

2 Additional analysis for motion relaxation

2.1 Relaxed warping loss for O.F. estimation

TOFlow [2] reveals that precise optical flow is not tailored for task-oriented applications, including frame interpolation. They have observed that precise optical flow does not lead to an optimal solution, especially for the occlusions. They address this problem by joint training the flow estimation and interpolation network without any constraints on the flow estimation.

In contrast, in our paper, we proposed a relaxed loss term for the flow estimation network to boost motion prediction as well as the final interpolation results. The effectiveness of the motion relaxation has been evaluated in the ablation studies by replacing it with $\mathcal{L}_{\ell_1}$. Here we perform an additional comparison of these cases with the model, which completely removes the loss function (No loss) for flow estimation, as reported in Table 1.

Table 1 shows the performance of our network trained with different warping loss function for the unsupervised flow estimation module on two datasets of Adobe240 and GOPRO. For all models evaluated in frame interpolation task, a flow estimation module is first trained using $\mathcal{L}_{\ell_1}$ as the warping loss, and then

Table 1: Investigating the impact of relaxation variations on Adobe240 and GOPRO dataset.

Loss for O.F.	Adobe240				GOPRO			
	PSNR	SSIM	IE	TCC	PSNR	SSIM	IE	TCC
$\mathcal{L}_{\ell_1}$	33.92	0.955	6.14	0.851	32.45	0.936	7.09	0.828
No loss	34.10	0.957	6.02	0.856	32.68	0.939	6.94	0.832
$\mathcal{L}_{w_{relax}}$	**34.37**	**0.959**	**5.89**	**0.860**	**32.91**	**0.943**	**6.74**	**0.837**

each model applies one of the loss functions in Table 1 for joint training with the rest of the network. The model trained without any constraint on O.F. [2], No loss, has better performance than the model trained with $\mathcal{L}_{\ell_1}$. However, it does not set a lower bound for the error that the flow estimation can tolerate. If the optical flow maps a pixel far away from its ground truth location as the reference frame, the performance for model prediction will be degraded as well as the final interpolation results.

In contrast, our proposed loss, $\mathcal{L}_{w_{relax}}$, provides the flexibility for the O.F. to move around the pixels only in the small neighborhood of their ground truth location. It accepts certain errors but also enforces a lower bound for the O.F. estimation constraint by limiting the window to a small neighborhood. Thus, the model trained with $\mathcal{L}_{w_{relax}}$ as the warping loss for O.F. estimation yields the best performance for final interpolation results.

2.2 Motion relaxation impact on warping, prediction and final results.

Table 5 in the main paper reports the effectiveness of $\mathcal{L}_{w_{relax}}$ by comparing the warped I_1 with corresponding ground truth. In Table 2, we show the comparison by warping I_0 as well as the final interpolation results for the middle frame. The results in Table 2 are consistent with Table 5 in the main paper, where with $\mathcal{L}_{w_{relax}}$, a better motion prediction and better final results are achieved.

Table 2: Motion relaxation evaluation in warping, middle frame prediction and final interpolation results. (warping from I_0)

Datasets	PSNR($I_0^{w\to 1}$, I_1)		PSNR($I_0^{w\to t_4}$, $I_{t_4}^{gt}$)		PSNR($\hat{I}_{t_4}$, $I_{t_4}^{gt}$))	
	$\mathcal{L}_{\ell_1}$	$\mathcal{L}_{w_{relax}}$	$\mathcal{L}_{\ell_1}$	$\mathcal{L}_{w_{relax}}$	$\mathcal{L}_{\ell_1}$	$\mathcal{L}_{w_{relax}}$
DAVIS	**29.47**	22.85	25.03	**25.48**	27.15	**27.91**

2.3 More visual examples to show the effectiveness of the motion relaxation

Figure 1 provides more samples from Adobe240 (first example) and DAVIS (second & third example) to show the impact of using $\mathcal{L}_{w_{relax}}$.

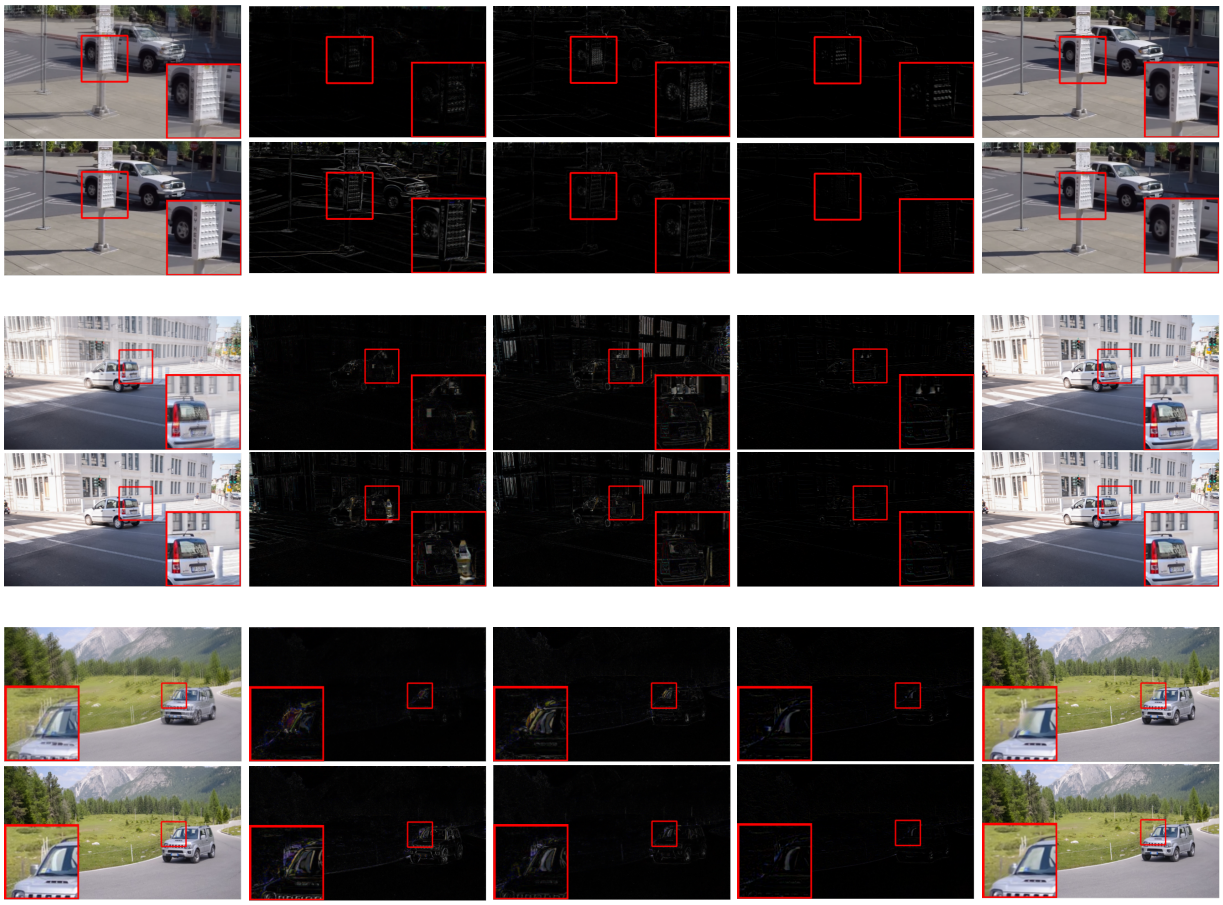

Figure 1: Comparison between O.F. estimation with/without (top/bottom row) relaxation in terms of the interpolation error for motion prediction and final interpolation result.

3 Architecture of the two-stages flow estimation

3.1 Network details for the flow estimation

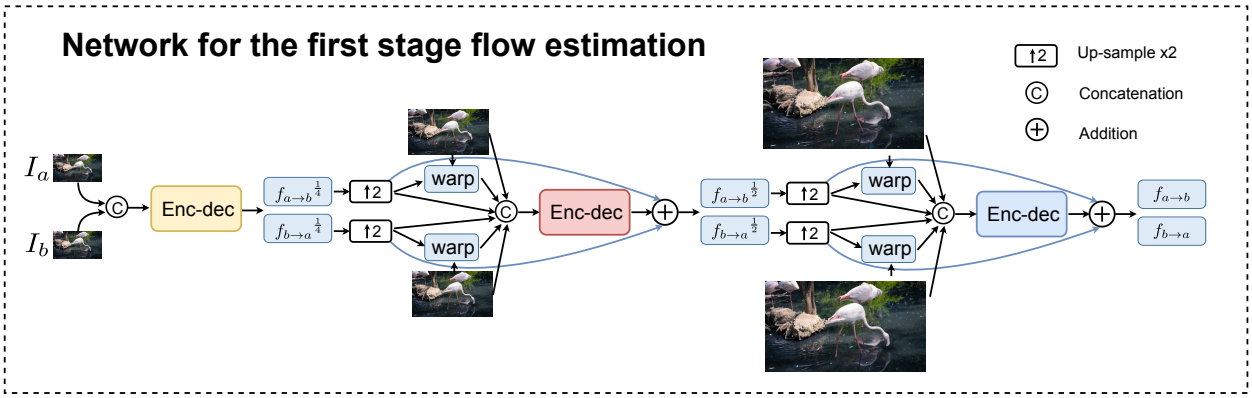

Figure 2: The coarse-to-fine architecture of the first stage flow estimation network where the first level (most left) estimate flow using 4x smaller resolution, the middle level refine flow estimation on 2x smaller resolution and final step work on original resolution

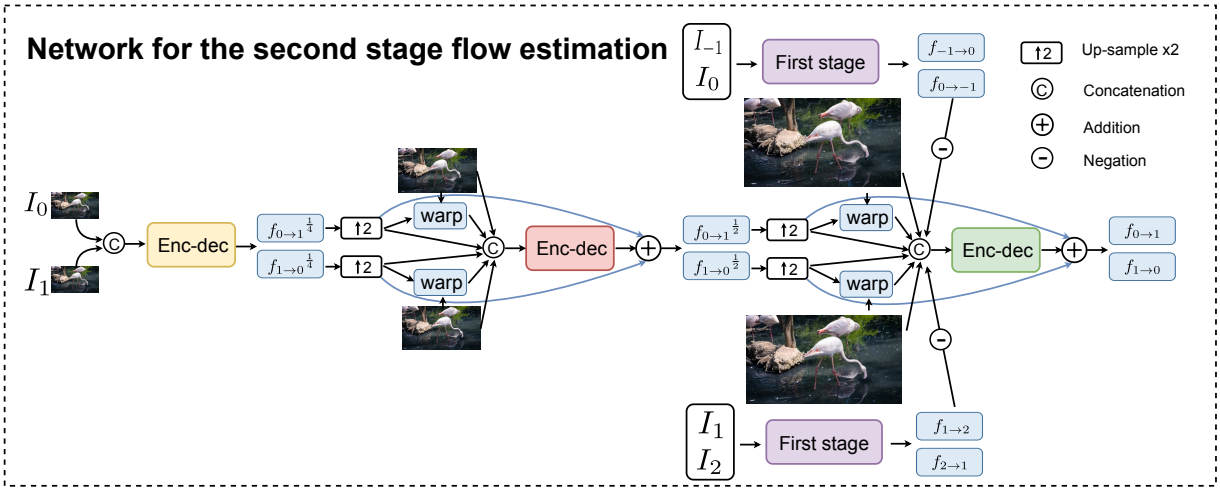

Figure 3: The second stage flow estimation network has the same structure as the network in the first stage. Except that we add $-f_{0\to-1}$ and $-f_{2\to1}$ to the input at the finest scale.

Since TOFlow [2] reveals that an accurate optical flow will result in sub-optimal performance of the interpolation results. Thus, instead of the heavy state-of-the-art flow networks [5, 4], we follow the light-weight coarse-to-fine flow estimation network in SPyNet [3]. We customize this network for efficiently computing the bidirectional O.F., and also benefiting from the multi-frame inputs.

Figure 2 shows the coarse-to-fine architecture for computing bidirectional O.F. between two input frames which is used in the first stage of our flow estimation network. The architecture of the second stage displayed in Figure 3 has the similar structure as the first stage. To benefit form I_{-1} and I_2 as the additional information, we first compute $f_{0\to-1}$ using I_{-1}, I_0 and $f_{2\to1}$ using I_1, I_2 using the first stage network. As $-f_{0\to-1}$ and $-f_{2\to1}$ are proper estimation of $f_{0\to1}$ and $f_{1\to0}$ respectively, we add them to the input of the finest scale (third level) in the second stage to help improve $f_{0\to1}$ and $f_{1\to0}$.

We employ the same encoder-decoder structure throughout the networks at both stages. Both stages

share the weights of two coarser levels (first and second levels), as indicated by the same color in Figure 2 and 3. The detailed architecture of the encoder-decoder structure will be provided in Section 3.2.

3.2 Encoder-decoder network

Table 4 reveals the details of the encoder-decoder network used in our flow estimation networks.

Table 3: Encoder-decoder network (activation function: LeakyReLU, with α=0.1).

	Input name	Output name	Kernel	Stride	Output spatial size
Input	-	concat_ims	-	-	$H \times W$
Encoder	concat_ims	Enc_conv_1	$3 \times 3 \times 16$	1	$H \times W$
	Enc_conv_1	Enc_conv_2	$3 \times 3 \times 24$	1	$H \times W$
	Enc_conv_2	Enc_conv_3	$3 \times 3 \times 24$	1	$H \times W$
	Enc_conv_3	Enc_ave_pool_1	-	2	$H/2 \times W/2$
	Enc_ave_pool_1	Enc_conv_4	$3 \times 3 \times 64$	1	$H/2 \times W/2$
	Enc_conv_4	Enc_conv_5	$3 \times 3 \times 64$	1	$H/2 \times W/2$
	Enc_conv_5	Enc_conv_6	$3 \times 3 \times 64$	1	$H/2 \times W/2$
	Enc_conv_6	Enc_ave_pool_2	-	2	$H/4 \times W/4$
	Enc_ave_pool_2	Enc_conv_7	$3 \times 3 \times 80$	1	$H/4 \times W/4$
	Enc_conv_7	Enc_conv_8	$3 \times 3 \times 80$	1	$H/4 \times W/4$
	Enc_conv_8	Enc_conv_9	$3 \times 3 \times 80$	1	$H/4 \times W/4$
Decoder	Enc_conv_9	Dec_conv_1	$3 \times 3 \times 64$	1	$H/4 \times W/4$
	Dec_conv_1	Dec_conv_2	$3 \times 3 \times 64$	1	$H/4 \times W/4$
	Dec_conv_2	Dec_conv_3	$3 \times 3 \times 64$	1	$H/4 \times W/4$
	Dec_conv_3	Dec_up_bilinear_1	-	-	$H/2 \times W/2$
	Dec_up_bilinear_1, Enc_conv_6	Dec_concat_1	-	-	$H/2 \times W/2$
	Dec_concat_1	Dec_conv_4	$3 \times 3 \times 64$	1	$H/2 \times W/2$
	Dec_conv_4	Dec_conv_5	$3 \times 3 \times 64$	1	$H/2 \times W/2$
	Dec_conv_5	Dec_up_bilinear_2	-	-	$H \times W$
	Dec_up_bilinear_2, Enc_conv_3	Dec_concat_2	-	-	$H \times W$
	Dec_concat_2	Dec_conv_6	$3 \times 3 \times 50$	1	$H \times W$
	Dec_conv_6	Dec_conv_7	$3 \times 3 \times 50$	1	$H \times W$
	Dec_conv_7	Dec_conv_8	$3 \times 3 \times 4$	1	$H \times W$

4 Temporal consistency

4.1 More visual results

Figure 4: Visualization of the seven intermediate frames of I_{t_1} to I_{t_7} generated by our method compared to Quadratic [1], SepConv [6], DAIN [8] and Super SloMo [7] from GOPRO. Our method generates more visually pleasant and temporally consistent frames. The main degradation is on the upper left and lower right of both inside and outside of the heart shape piece.

4.2 Explain the curves

Figure 5 is the same figure of Figure 9(c) in the main paper. All the methods showed in this figure follow the similar trend of the expected quality of the intermediate frames as a function of the temporal distance to the input frames. However, DAIN [8] is one exception with a local peak in the middle frame. The reason is that, during training, DAIN is interpolating only the middle frame by setting the time stamp $t_i = 0.5$. However, at inference time, they interpolate multiple frames by setting t_i to different values. Thus, it has relatively better performance at $t_i = 0.5$. In contrast, Quadratic [1] and Super SloMo [7] are trained with frames at different time stamps t_i, and they follow the common trend of quality in generating multiple frames.

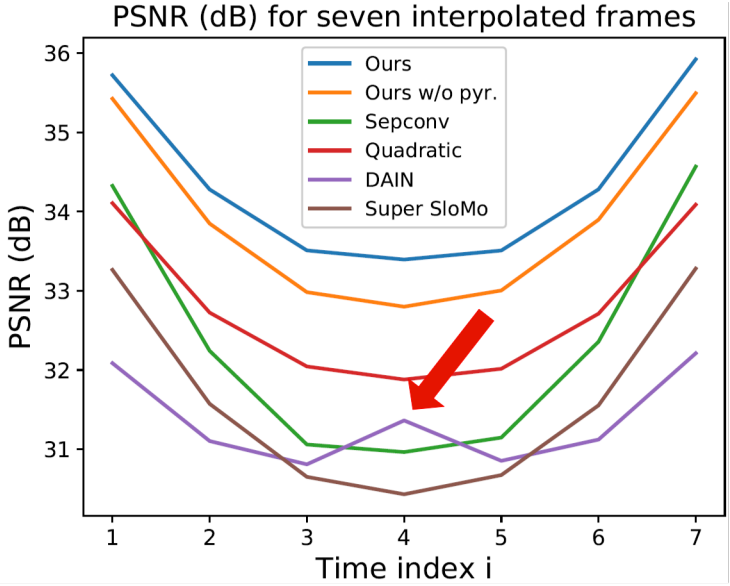

Figure 5: Performance trend in multiple frame interpolation.

5 Explain on evaluation of Quadratic

Table 4: Performance evaluation of the proposed method compared to the state-of-the-art methods in different datasets.

Methods	Adobe240			GoPro			Vimeo90K			DAVIS		
	PSNR	SSIM	TCC	PSNR	SSIM	TCC	PSNR	SSIM	IE	PSNR	SSIM	IE
Quadratic(original paper 960fps data)	32.95	0.966	-	31.27	0.948	-	-	-	-	27.72	0.894	12.32
Quadratic(our 240fps data)	32.80	0.949	0.842	32.01	0.936	0.822	33.62	0.946	5.22	27.38	0.834	12.46
Ours	34.37	0.959	0.860	32.91	0.943	0.837	34.93	0.951	4.70	27.91	0.837	12.40

The reason of causing the discrepancy between the numbers reported by us and Quadratic is that the original model was trained on 960fps videos which is not publicly available. For fair and faithful comparison, we retrained the Quadratic model on our training data (240fps). We strictly followed the source code and training procedure provided. The 960fps training data was also down-sampled to 240fps and 480fps which provides different training data distributions. As for SSIM values, they can be different because of the parameters of SSIM or the software used (MATLAB yields different values compared to Python). To be fair, we followed the same SSIM evaluation code used in Super SloMo throughout the paper.

6 More visual examples

Figure 6, 7, 8 are the sample results of challenging cases. We provide the videos (Compare_with_SOTA.mp4) to better show the comparison with other methods in terms of visual quality. In addition, to show the capability of our multi-frame interpolation method in terms of maintaining the temporal consistency, we generate another video (Temporal_consistency_60_to_1920fps.mp4) with interpolating a different number of frames, from 1 frame to 63 frames results in videos of 60fps to 1920fps from a 30fps input video.

(a) Input (b) Super SloMo [7] (c) Quadratic [1] (d) Ours

Figure 6: A visual example from DAVIS dataset. In this example, it contains complex motion, such as highly non-linear and large motion. The leg of the flamingo is very thin, which brings the difficulties to estimate the optical flow. Our method handles this situation well compared to Super SlowMo and Quadratic.

(a) Input (b) Super SloMo [7] (c) Quadratic [1] (d) Ours

Figure 7: A visual example from GOPRO dataset. It contains severe camera movements (changing in direction frequently). It is challenging to interpolate consistent multiple frames at the moment when the camera suddenly changes the direction. Our one-shot pyramidal structure yields better results.

(a) Input (b) Super SloMo [7] (c) Quadratic [1] (d) Ours

Figure 8: A visual example from DAVIS dataset. In this example, it contains the combination of translational motion, which involves non-constant accelerations. Our methods with advanced motion modeling is able to handle this situation well.

7 Additional analysis on cubic motion modeling

The performance of the cubic motion model is evaluated in Table 3 for frame interpolation tasks, and also Fig. 2 shows a comparison for a simulated motion scenario.

Further evaluation can be performed on motion prediction for non-rigid objects with complex motions in real scenarios in which human body motions are a good fit to this end. We evaluated our cubic motion modeling in Eq. 5 of the paper and (13) in predicting motions of landmarks on the human body using a comprehensive set of captured motions by MoCap techniques. We specifically used the CMU Graphics Lab Motion Capture Database [9], which has 109 subjects each has done various motions such as walking, dancing, jumping, and kicking ball. The motions are captured using 12 Vicon infrared MX-40 cameras in 120 fps rate. Motions are captured for the variable number of landmarks in different captures with at least 41 landmarks in each session. As the captured motions have different lengths, we made non-overlapping patches of 7 consecutive records from a landmark for our motion analysis. In each path of seven samples, we considered samples 1, 3, 5, and 7 as consecutive observations used in motion models and tried to predict landmark position between samples 3 and 5 by using 4 as the ground truth. We used both quadratic and cubic models for prediction and measured the absolute difference of predicted position as an error metric. Fig. 9 shows the performance comparison and it reveals that the cubic modeling works better in predicting complex motion.

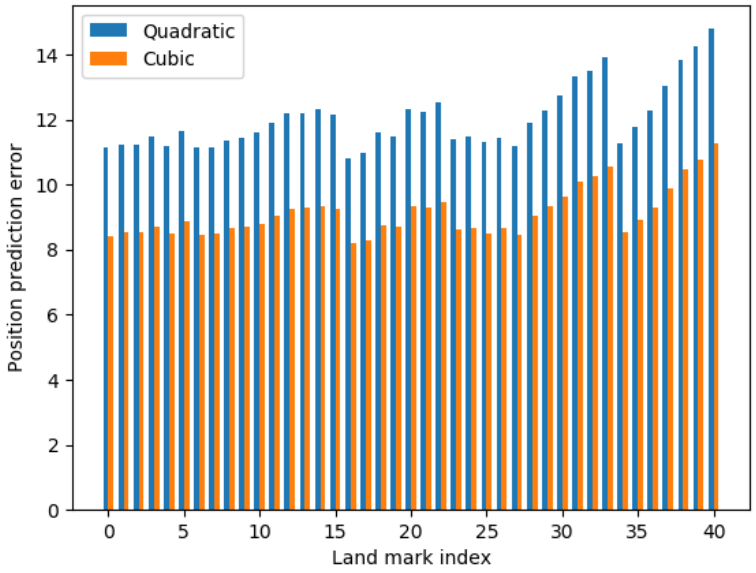

Figure 9: Performance of motion prediction models (quadratic vs. cubic) on MoCAP data from CMU dataset. The prediction error is the absolute difference in the actual position and predicted one.

[1] Xu, Xiangyu and Siyao, Li and Sun, Wenxiu and Yin, Qian and Yang, Ming-Hsuan Quadratic video interpolation. In *Advances in Neural Information Processing Systems*, 2019.

[2] Xue, Tianfan and Chen, Baian and Wu, Jiajun and Wei, Donglai and Freeman, William T Video enhancement with task-oriented flow. In *International Journal of Computer Vision*, 2019.

[3] Ranjan, Anurag and Black, Michael J Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[4] Ilg, Eddy and Mayer, Nikolaus and Saikia, Tonmoy and Keuper, Margret and Dosovitskiy, Alexey and Brox, Thomas Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[5] Sun, Deqing and Yang, Xiaodong and Liu, Ming-Yu and Kautz, Jan Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[6] Niklaus, Simon and Mai, Long and Liu, Feng Video frame interpolation via adaptive separable convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[7] Jiang, Huaizu and Sun, Deqing and Jampani, Varun and Yang, Ming-Hsuan and Learned-Miller, Erik and Kautz, Jan Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[8] Bao, Wenbo and Lai, Wei-Sheng and Ma, Chao and Zhang, Xiaoyun and Gao, Zhiyong and Yang, Ming-Hsuan Depth-aware video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[9] CMU Graphics Lab Motion Capture Database http://mocap.cs.cmu.edu/ Online; accessed 05-Mar-2020.