Feature Flow for Frame Interpolation

Ricson Cheng, Rishub Jain, Hariank Muthakana
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
{ricsonc,rishubj,hmuthaka}@andrew.cmu.edu

Abstract

Frame interpolation is challenging task because it requires an understanding of motion. A good solution to this problem would result in a better understanding and representation of object motion. Inspired by recent approaches which use optical flow between consecutive frames, we present another flow based solution. However, reasoning about motion at a pixel level can be challenging because of occlusions and noise. We argue that motion prediction could be improved by learning in a more abstract feature space. Our proposed architecture uses a frame autoencoder to learn an abstract representation of images along with a feature flow network to predict motion in this abstract space.

1 Introduction

1.1 Frame Interpolation

Frame interpolation is the task of predicting the image $\hat{I}_t$ between two frames I_{t-1} and I_{t+1}. Given an input video $I_1 \dots I_T$, interpolating between every pair of consecutive frames would double the frame-rate.

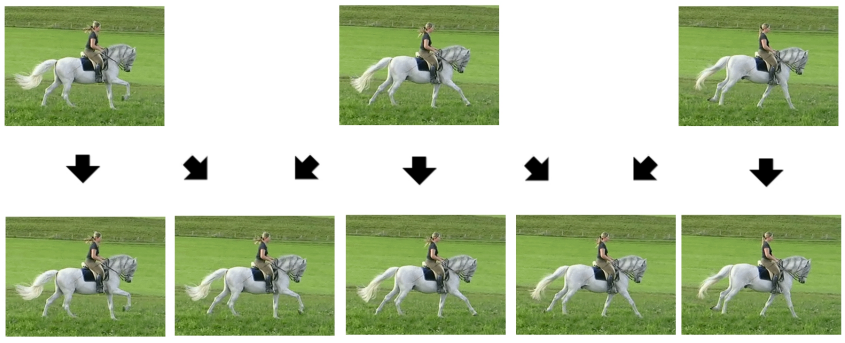

Figure 1: Interpolation of a video of a horse moving. [12]

1.2 Motivation

The most immediate application of frame interpolation is for use in video smoothing. Since frame interpolation can be used to predict the frame which "goes between" two other frames, applying frame interpolation between every consecutive pair of frames in a video will double the number

of frames, therefore doubling the rate at which frames are displayed. This increases the perceived smoothness of the video.

More theoretically, frame interpolation is interesting because explores motion. Videos are mostly composed of objects whose appearance does not change. The content of a video is in the motion of these objects. For example, we understand that a person does not usually change appearance. If a person turns around, hiding their face, we understand that their face has not changed appearance – it has merely moved somewhere beyond our sight.

Our proposed solution to frame interpolation utilizes this notion of appearance and motion. This understanding of motion is something which can be transferred to many other problems in computer vision such as video prediction, trajectory prediction, and egomotion estimation. Therefore, our appraoch will be of interest to practitioners in a wide range of areas.

2 Background

2.1 Optical Flow

The optical flow F of an image indicates pixel correspondences between two images. For example, if the corner of a box is at pixel $(3, 3)$ in one image, and $(5, 6)$ in the next, then the flow would be $(1, 3)$. The flow map is therefore a tensor with shape $(H, W, 2)$. The flow map can be used to move the pixels in an image to the location prescribed by the flow. This is known as warping.

Define I^i, F, and I^o be the input image, input flow, and output image of the warping function respectively. The warping is defined as:

$$\texttt{warp}(I^i, F)_{ij} = \texttt{bilinear}(I^i, (i, j) + F_{ij}) \tag{1}$$
$$= I^o_{ij} \tag{2}$$

Where `bilinear` is the standard bilinear interpolation algorithm for sampling the color value of a discrete image at floating point coordinates. This is necessary to maintain the differentiability of the warping function.

2.2 Related Work

The simplest approach to frame interpolation is frame averaging, which simply averages the two adjacent frames. This usually produces poor results, but is a good baseline.

Previous works by Xiao et al., and Yahia et al. have simply predicted the interpolated frame by passing the image pair into a CNN and predicting the interpolated frame. This naive approach often produces blurry output and does not seem to perform substantially better than frame averaging.

Samsonov also experimented with predicting the optical flow, and then using the optical flow to produce the interpolated image. This approach has the advantage of producing sharper images, since the network has the relatively easy task of moving pixels instead of predicting them.

Meyer et al. use an eulerian approach to interpolation. The eulerian approach models the change in pixel color instead of pixel motion. Their approach is computationally efficient and compares favorably to optimal flow based methods, especially in scenes with high contrast. However, it is not learning based, which means the model can't improve itself by watching more videos.

Niklaus et al. uses a highly specialized CNN architecture to perform frame interpolation. Their CNN takes as input a region of the image pair in order produce a kernel, which is convolved with a patch of the image pair, producing an output pixel. They claim that this approach improves over traditional optical flow methods by combining both the pixel correspondence step and the pixel synthesis step into one.

Amersfoort et al. is a very recent paper which takes a similar approach to ours. They predict flow at multiple scales, with each successive level of flow "refining" the estimates of the previous levels. In addition, they take advantage of GANs in order to improve visual quality. The difference between this work and ours is that they use multi-scale flow to warp just the input image, whereas we use the flows to warp feature maps instead.

Our approach is based off the optical flow technique used by Samsonov. We claim that the naive optical flow approach does not perform well because of occlusions and noise. Occlusions are instances when one object moves in front of another object, occluding the other object and hiding the motion of the other object. Small objects are more easily occluded than large objects. Since optical flow deals with motion on the pixel level, and pixels are very small and easily occluded, occlusions make optical flow unreliable when it comes to modeling motion.

3 Methods

A human, when observing a moving car, will see the entire car as one object, and not as a large collection of points which are all moving in the same direction. This suggests that humans reason about motion in an feature space which contains complex concepts such as "car". Our approach essentially mimics this aspect of human perception by lifting motion prediction into this feature space.

3.1 Feature Flow Network

As shown in Figure 2, our network consists of a frame autoencoder and a flow network. The frame autoencoder follows a typical encoder-decoder structure with skip-connections (not pictured). The flow network takes as input two consecutive frames and predicts the flow using an encoder-decoder structure. However, the flow network is unusual in that it predicts a flow map at every resolution. These flowmaps are used to warp the features at each stage of the frame decoder.

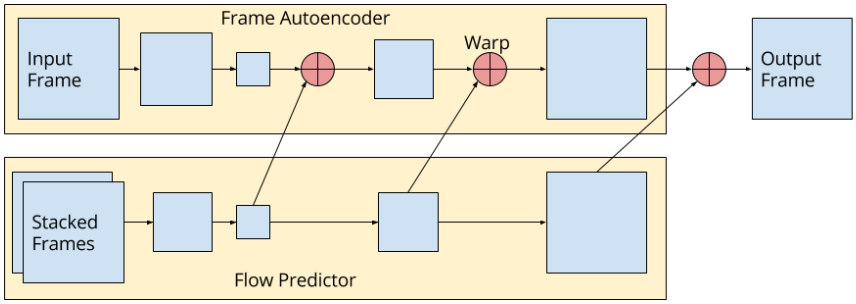

Figure 2: Both the flow network and the frame autoencoder have fully convolutional architectures. The last two channels of every feature block in the flow network are extracted as the flow map at that resolution.

The job of the frame autoencoder is to learn high level features of frames from a video in an unsupervised manner. It utilizes a standard encoder-decoder architecture. The frame autoencoder can operate in two modes – either in the typical autoencoder mode, or in *prediction mode*, where it uses inputs from the flow network.

The job of the feature flow network, which has a similar encoder-decoder architecture, is to produce the flow maps. However, in contrast to typical flow networks which predict the optical flow of pixels in an image, this model attempts to predict the optical flow on the feature maps produced by the frame autoencoder. The key novelty of our approach is to shift the prediction of motion from the image space to the feature space.

The decoder portion of the frame autoencoder produces a sequence of feature maps G_i, with each successive feature map having twice the size of the last. The last feature map is then convolved to produce the final output frame. We can write this as:

$$G_i = f_i(G_{i-1}, \theta_i) \tag{3}$$

where θ_i is the weights in the ith layer of the decoder, and f_i is one layer of the decoder.

Simultaneously, the frame predictor produces feature maps H_i in the same manner. The feature flow is extracted by taking the last two channels of every feature map H_i, giving us flow maps F_i. In order to run the decoder in *prediction mode* with the flow, we modify the equation to be:

$$G_i = \texttt{warp}(f_i(G_{i-1}, \theta_i), F_i) \tag{4}$$

Under this prediction mode, the output of the frame autoencoder is taken to be the interpolated frame that we want.

3.2 Training

There are two terms to minimize during training: the reconstruction loss and the prediction loss.

For each training triplet: I_{t-1}, I_t, I_{t+1}, the first and last frame I_{t-1} and I_{t+1} are fed through the autoencoder with no interaction from the flow network. The autoencoder then produces $\hat{I}_{t-1}$ and $\hat{I}_{t+1}$. The reconstruction loss is:

$$\mathcal{L}_r = J(I_{t-1}, \hat{I}_{t-1}) + J(I_{t+1}, \hat{I}_{t+1}) \tag{5}$$

where in practice, the loss function J is L1 loss.

Next, we feed I_{t-1} through the frame encoder, and I_{t-1}, I_{t+1} through the flow network. This time, the flow maps from the flow network are used to warp features in the decoder, producing $\hat{I}_t$. So the prediction loss is:

$$\mathcal{L}_p = J(I_t, \hat{I}_t) \tag{6}$$

In addition, we add on some regularization terms, such as a L2 weight decay term, and a smoothness loss on the predicted flows. Denote the regularization loss terms as $\mathcal{L}_g$. This gives us the total loss.

$$\mathcal{L}_{\text{total}} = \alpha\mathcal{L}_r + \beta\mathcal{L}_p + \mathcal{L}_g \tag{7}$$

The smoothness loss on the predicted flows reflects our priors that objects in a video move coherently. Adjacent pixels tend to move in the same direction, and we might expect this holds true even for higher level features.

4 Experiments

4.1 Datasets

We use consecutive image frame triplets from the MPI Sintel dataset, which is an animated film commonly used to benchmark optical flow. It is a challenging dataset because it contains large motions and fast movements. The dataset contains 21310 frames.

We also used frames from a subset of the KITTI dataset, which is a dataset captured by a camera mounted on a car driving around in a city. This dataset is real data, as opposed to the synthetic Sintel dataset. Since the domain is very different from the Sintel dataset, it can be used for challenging domain generalization experiments. We used a subset of the dataset containing 41472 frames.

4.2 Baseline Models

As a baseline to compare against, we use image pair averaging: $\hat{I}_t = \frac{1}{2}(I_{t-1} + I_{t+1})$.

In addition, we compare against models from previous works. For our first baseline, we predict the difference from the average frame to the true interpolated frame, I_t, using an encoder-decoder

architecture which takes the stacked image pair as input. We call this model the direct prediction network.

We also evaluate against the flow network from Samsonov, but using our own network architecture. We first predicting the optical flow from the first frame to the interpolated frame, and then apply this flow on the first frame. This optical flow is predicted with the same encoder-decoder architecture. We call this model the flow prediction network.

4.3 Experimental Details

4.3.1 Model Architecture

Our architecture is very similar to both U-Net, a fully convolutional architecture which has previously performed well in flow prediction and image segmentation tasks [8], and also Feature pyramid networks, which have been used for segmentation and object detection [9]. Crucially, both of these architectures aim to capture the strong semantics of high level features while integrating the weaker semantics, but higher precision of lower level features.

For all experiments, we reuse the same encoder-decoder architecture. The input of the encoder and output of the decoder were of equal size. Both the encoder and the decoder consist of an equal number of *blocks* – we used 4 blocks for all of our experiments. Each block in the encoder doubles the number of channels, and halves the height and width of the feature map. Each block in the decoder halves the number of channels and doubles the height and width of the feature map. Before the first encoder block is a convolution which outputs a full-size 16 dimensional feature map. After the last decoder block is a convolution which outputs a full-size k dimensional feature map, where k would be 2 for flow prediction and 3 for image prediction.

Each of the encoder blocks consist of two convolutions, and then a 2 by 2 pooling layer. Each decoder block makes a prediction with dimension k by convolving the input feature map. It then upscales the input feature map, puts another convolution, then concatenates all of the upscaled prediction, the convolved feature map, a feature map of the same size from the corresponding encoder block (a skip connection). All convolutions are 3 by 3 with stride 1 and no dilation. Batch normalization is used after every layer. Weight decay was used. Each layer was followed by a relu activation. We used upscaling rather than deconvolution to avoid the checkerboard artifacts [7].

For the frame autoencoder, we removed the skip connections between the lower-level feature maps in order to encourage the autoencoder to learn better features, and not copy the input frames.

4.3.2 Hyperparameters

For both the direct prediction and direct optical flow networks, we used a batch size of 8. For the feature flow network, we used a batch size of 2 due to its higher memory usage. During training, the loss converges at around 750K iterations, although we trained for only 300K due to limited computational resources. We used the ADAM optimizer with a learning rate of 0.0002 and a momentum of 0.9. For our loss, we used $\alpha = 10$ and $\beta = 20$. Due to limited resources, we did not perform any hyperparameter search.

4.3.3 Dataset Preprocessing

We downscaled the images so that we could test our experiments with more limited computational resources. We resized the Sintel dataset images to $(272, 480)$ and the KITTI dataset images to $(192, 656)$. We normalized the RGB values of each image to the range $(0, 1)$. No further data augmentation was performed.

For each dataset, we partitioned the frames into train, validation, and test datasets. This was done temporally (so all frames in each subset are consecutive to each other) so that our evaluation would be more sensitive to overfitting. For example, this temporal split prevents our models from performing well simply by memorizing the first half of the dataset, because validation is done on frames at the end of the dataset, which will necessarily contain different scenes with very different objects.

4.4 Experiments and Results

In all tables, we report the average L1 loss between I_t and $\hat{I}_t$ across all frames in the evaluation set.

The first experiment was to train and test on the MPI Sintel dataset, using our disjoint train and test set. We did the same on KITTI. 1.

Table 1: Interpolation experiments

Model	Sintel	KITTI
Image Pair Averaging	0.0163	0.00661
Direct Prediction	0.0166	0.01382
Flow Prediction	0.0131	0.00659
Feature Flow	0.0241	0.00857

Next, we performed a cross-domain generalization experiment, training on Sintel and testing on KITTI and vice versa 2.

Table 2: cross-domain generalization experiments

Model	Sintel to KITTI	KITTI to Sintel
Image Pair Averaging	0.00661	0.0163
Direct Prediction	0.00687	0.0248
Flow Prediction	0.00754	0.0148
Feature Flow	0.01553	0.0172

In addition, we performed a "long-range" experiment to test how well each method performs when the motion is extremely large. To do this, we predicted I_t from frames I_{t-k} and I_{t+k}. In the usual case, $k = 1$. Here, we set $k = 4$ and $k = 8$ 3.

Table 3: long-range experiments

Model	$k = 4$	$k = 8$
Image Pair Averaging	0.0439	0.0664
Direct Prediction	0.0437	0.0630
Flow Prediction	0.0437	0.0696
Feature Flow	0.0549	0.0804

4.5 Qualitative Results

To qualitatively compare our methods, we test on the long-range ($k = 8$) experiments which have high amount of motion. Figures 3-5 show long-range experiment outputs on a specific triplet. Both frame averaging and direct interpolation produce blurry outputs, as can be seen in the first and second images of Figure 4.

Flow prediction reconstructs the objects in the scene better, but at the cost of severe distortion, as can be seen by comparing the results of flow (third image in Figure 4) with the results of feature flow (middle image in Figure 5). It can be seen that flow prediction distorts the pattern of tiles on the ground, while feature flow produces the most aesthetically pleasing interpolation.

5 Conclusion

On the first set of experiments 1, we see that the flow prediction method does better than direct prediction, which has been confirmed by previous works. We also found that direct prediction does not significantly improve over the frame averaging baseline. Finally, the feature flow method does not perform very well compared to the other methods.

Figure 3: The ground truth image triplet from the long range ($k = 8$) experiment

Figure 4: The results of the different methods from left to right: Image Pair Averaging, Direct Prediction, Flow Prediction, the predicted flow map from the flow prediction network

Figure 5: The autoencoded images from the Feature Flow network of each frame of the image triplet. The middle image is the output interpolated frame.

In the generalization experiments 2, the feature flow method performs relatively well when going from KITTI to Sintel, but poorly when trained on Sintel and tested on KITTI. In terms of percent-increase in error, it seems that all three methods do about equally well, although there is too much variability in the results to be sure.

We expected feature flow to generalize better than other methods, but it did not happen. We suspect this is because the autoencoder in the feature flow method failed to generalize across domains. We expect a flow network to generalize better than an autoencoder because motion is usually more similar between domains than appearance. In order to solve this issue, one possible approach would be to fine-tune the autoencoder on the new domain at test-time, and then run the full feature flow network to make predictions.

For the long-range experiments 3, we found that direct prediction performed surprisingly well, and feature flow performed slightly worse. However, in terms of measuring how performance degrades with larger k, the L1 loss of the direct prediction and flow prediction method increased by a factor of 3.8 and 5.3 times respectively from $k = 1$ to $k = 8$, while the loss of the feature flow method increased by only 3.3. In addition, we found that qualitatively, the results of feature flow tend to be quite good compared the other three methods.

5.1 Future Work

One deficiency of the feature flow approach is the tension between low-level precision and high-level feature learning. Providing low-level skip connections to the autoencoder ensures that it is capable of producing sharp outputs, which is valuable. On the other hand, low-level skip connections may make it too easy for the autoencoder to copy frames without substantial learning. This prevents good high-level features from being learned, which prevents good prediction. Removing low-level skip connections would solve this problem, but cause the output to be far more blurry. In future work, we may explore possible solutions to this problem.

One advantage of using optical flow to warp images is that the output is very sharp. This is due to the fact that the network does not need to make sharp predictions. The predicted flow is merely used to move pixels around from the input image, which is already sharp. The autoencoder which is

necessary to our frame flow network causes us to lose this sharpness advantage. In the future, we hope to explore possible ways to keep our predictions sharp.

References

[1] Meyer, S., Wang, O., Zimmer, H., Grosse, M., & Sorkine-Hornung, A. (2015). Phase-based frame interpolation for video. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1410-1418).

[2] Niklaus, S., Mai, L., & Liu, F. (2017). Video Frame Interpolation via Adaptive Convolution. *arXiv preprint arXiv:1703.07514*.

[3] Samsonov, V. (2017). Deep Frame Interpolation. *arXiv preprint arXiv:1706.01159*.

[4] Yahia, H. B., Intelligentie, K., & Reisser, M. (2016). Frame Interpolation using Convolutional Neural Networks on 2D animation (Doctoral dissertation, MA thesis. University of Amsterdam).

[5] Xiao, T., Puri, R., & Kesineni, G. Frame Rate Upscaling with Deep Neural Networks.

[6] van Amersfoort, J., Shi, W., Acosta, A., Massa, F., Totz, Johannes, Wang, Z., & Cabellero, J. (2017). Frame Interpolation with Multi-Scale Deep Loss Functions and Generative Adversarial Networks *arXiv preprint arXiv:1711.06045*.

[7] Odena, A., Dumoulin, V., & Olah, C. (2016). Deconvolution and Checkerboard Artifacts. Distill.

[8] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv preprint arXiv:1505.04597*

[9] Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2016). Feature pyramid networks for object detection. *arXiv preprint arXiv:1612.03144*.

[10] Butler, D. J., Wulff, J., Stanley, G. B., & Black, M. J. (2012). A Naturalistic Open Source Movie for Optical Flow Evaluation. Ê*uropean Conference on Computer Vision* (pp. 611-625).

[11] Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research*.

[12] File:Motion interpolation example.jpg. (2016, May 26). *Wikimedia Commons, the free media repository*.