Video Deblurring with optical flow Estimation CS231a Project Report

Suvasis Mukherjee
Stanford University
(suvasism@stanford.edu)

I. *Abstract*

Videos captured by smart phones or handheld cameras are vulnerable to motion blur due to camera shakes and object motion. Video deblurring increases the visual quality of a video, and can also improve the accuracy of other video processing tasks, such as object recognition, tracking, and 3D reconstruction. Video deblurring methods[1] typically rely on information about motion between neighboring frames, because it can be used for roughly estimating the blur and aligning frames to provide different captures of the same scene

II. *Introduction*

To estimate motion precisely is difficult in presence of blur. Blur incorporates inaccurate estimate of motion deformation in motion compensation process. In this project I have adopted LiteFlowNet[9], FlowNet2[7] and Pixel Volume[3] for motion compensation. LiteFlowNet[9] is a modified version of FlowNet2[5], it is trained with a blur-invariant loss so that the trained network can estimate blur-invariant optical flow between frames. LiteFlowNet[9] uses frame wrapping (ref fig.3) for motion compensation. Aligning a frame by warping Frame introduces inaccuracies which can effect deblurring process. Hence in this project I adopted pixel volume[9]. Pixel volume consists of multiple matching candidates for each pixel. The video deblurring framework uses a recurrent CNN structure. The framework take 4 inputs: the previous, current, and next blurry input frames, and the deblurred result of the previous frame. The framework then estimates motion between the current and previous frames in a blur-invariant way by using LiteFlowNet[9], then uses the estimated motion to construct a pixel volume from the deblurred previous frame. Finally, the deblurring network restores a sharp image for the current frame using the pixel volume with input blurry frames.

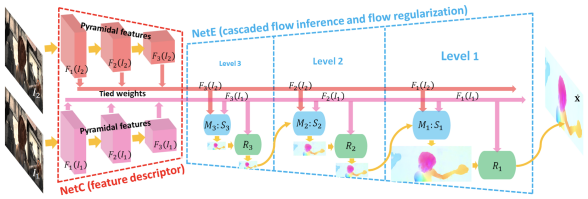
III. *Method*

A goal of this project is to employ blur-invariant learning to estimate optical flow maps between sharp video frames and use them as ground truth labels for training optical flow estimation between blurry frames.

Optical Flow using liteFlowNet

I have adopted the network architecture from, Recurrent Video Deblurring with Blur-Invariant Motion Estimation and Pixel Volumes[3]. It uses the base model of LiteFlowNet[9] [Hui et al. 2018] with some modifications borrowed from lite-FlowNet3[10]. LiteFlowNet is composed of two compact sub-networks that are specialized in pyramidal feature extraction and optical flow estimation as shown in Figure 1. The 2 sub-networks are NetC and NetE. NetC (Siamese network structure) transforms any given image pair (I1 , I2) into two pyramids of multi-scale high-dimensional features and a co-relation layer finds match between features of the 2 images. NetE consists of cascaded flow inference and regularization modules that estimate coarse-to-fine flow fields.
Video deblurring methods typically rely on information about motion between neighboring frames, because it can be used for roughly estimating the blur and aligning frames to provide different captures of the same scene.

Fig. 1. LiteflowNet Network.



Accurate estimation of motion is difficult in the presence of blur, and incorrectly estimated motion may cause structural deformations during the motion compensation process, and degrade the deblurring quality. First, to estimate motion accurately paper adopted LiteFlowNet[9] [Hui et al. 2018] and train it with a blur-invariant loss so that the trained network can estimate blur-invariant optical flow between frames.

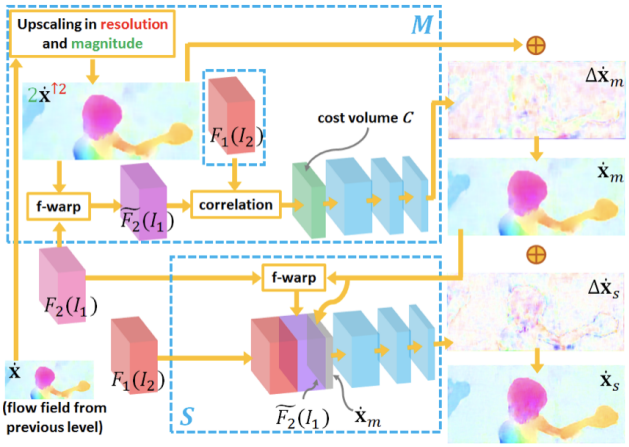
NetC is a two-stream network in which the filter weights are shared across the two streams. Each of them functions as a feature descriptor that transforms an image I to a pyramid of multi-scale high-dimensional features $F_k(I)$ from the highest spatial resolution $(k = 1)$ to the lowest spatial resolution $(k = L)$.

$$F\ (x) = \Sigma_{x_s^i} F(x_s^i(1 - |x_s - x_s^i|))(1 - |y_s - y_s^i|)$$

where $x_s = x + x^{\cdot} = (x_s, y_s)^T$ is the sample point and F defines input feature map. F is the interpolated feature map, $N(x_s)$ denotes 4 pixel neighbors of x_s. Flow

regularization, estimated flow field can be fragile to outliers if data fidelity is used alone. Module R in NetE regularizes flow field by adapting the regularization kernel through a feature-driven local convolution (f-lcon) layer.

Fig. 2. LiteflowNet Inference (NetE) .



A cascaded flow inference module M:S in NetE.

Cascaded flow inference, at each level of NetE, pixel-by-pixel matching (M) of high-level features yields coarse flow estimate. A subsequent refinement (S) on the coarse flow further improves it to sub-pixel accuracy.

$$c(x,d) = F1(x) \cdot F2(x+d)/N$$

where c is the matching cost between point x in $F1$ and point $x+d$ in $F2$, $d \epsilon Z$ is the displacement vector from x, and N is the length of the feature vector. A cost volume C is built by aggregating all the matching costs into a 3D grid. Feature warping, to facilitate large-displacement flow inference, the high-level features of the second image is warped towards the high-level space of the first image by a feature warping (f-warp) layer at each pyramid level.

$$F_g(c) = G\ (c) \odot F\ (c)$$

G is the regularization filter applied to feature F in the f-lcon layer.

Pixel Volume

The effective motion compensation can be done by aligning a frame by warping(ref fig3). In this project the motion compensation is done generating a pixel volume[3](ref fig 4) that consists of multiple matching candidates for each pixel.

First, liteFlowNet[9] estimates optical flow W_{t-1} from frame I_t^b to frame I_{t-1}^b and estimate the frame I_{t-1}^{est}.

Fig. 3. Feature warping to facilitate large-displacement flow inference, the high-level features of the second image is warped towards the high-level space of the first image by a feature warping layer at each pyramid level.

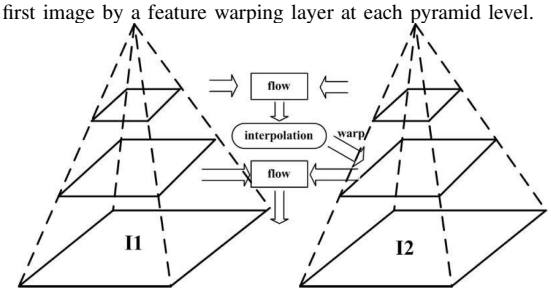
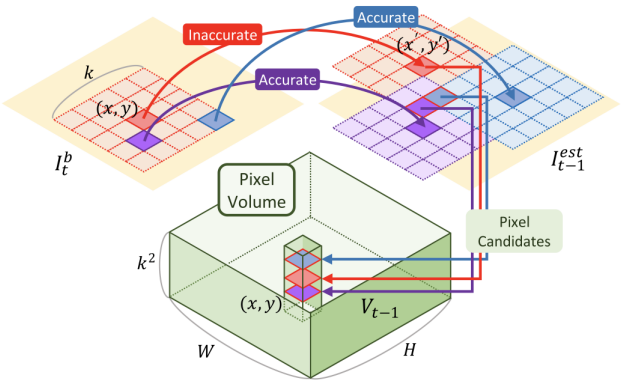


Fig. 4. Pixel Volume .



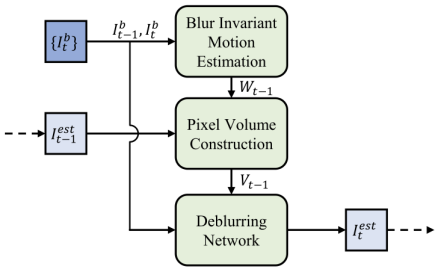
Second, Pixel volume[3] generation, ref, fig. 4, for pixel (x,y) of I_t^b consider a special window of size kxk centered at (x,y). The window size k can be 1,2,3,4,5,6,7. I have tested with k=5 and k=7. larger window size, k=7 is computationally expensive and it takes more time. For k=5 results are better compared to other window size. For each pixel, $(x+\delta x, y+y))$ in the window, where $\Delta x, \Delta y$ in $-k/2, ..., k/2$. The matched pixel (x', y') in I_{t-1}^{est} is determined by the optical flow W_{t-1}. Then the neighboring pixel $(x' - \delta x, y' - \delta y)$ in I_{t-1}^{est} is a matching candidate for pixel (x,y). Hence, there are k^2 matching candidates for I_{t-1}^{est} in (x,y). Now collect such matching candidates for all pixels I_t^b and stack them to obtain a pixel volume V_{t-1} of size $WXHXh^2$, where W and H are the width and height of a video frame.

Combine 3 consecutive input blurry frames, I_{t-1}^b, I_t^b, I_{t+1}^b and deblurred previous frame, I_{t-1}^{est} as input frames. From the LiteFlowNet described above which takes two images, $I_t^b and I_{t-1}^b$ and generates optical flow between them, W_{t-1}. Use this optical flow W_{t-1} and the input frames I_{t-1}^{est} from the previous deblurred frame to the framework's pixel volume generator and creates V_{t-1}, pixel volume.

Finally input, I_{t-1}^b, I_t^b, I_{t+1}^b to the pixel volume, V_{t-1} to produce the deblurred I_t^{est}

Feed the pixel volume V_{t-1} and the input frames,I_{t-1}^b, I_t^b, I_{t+1}^b, to the encoder - decoder framework to reconstruct

Fig. 5. Video Deblurring Framework



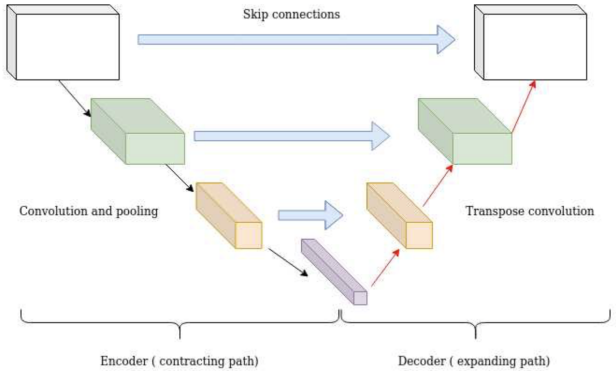
resulting framework. Input is divided into two groups, the concatenated consecutive frames, and the pixel volume. The network has a long skip-connection from the input frame to the network output to make the network predict only residuals while preserving overall contents of the input frame.

To train Video Deblurring Framework, we use mean absolute difference between deblurred frames and their corresponding ground truth sharp frames as the loss function.

Detailed Architecture of Video Deblurring Framework

Feature Transformation Layers

Fig. 6. video deblurring framework uses much of of the architecture from U-Net[14]



Layer type	input	size	strike	out	reg.	acti
conv	V_{t-1}	3x3	(2,2)	64	BN	relu
conv	64	3x3	(2,2)	64	BN	relu
conv	64	3x3	(2,2)	64	BN	relu
sum1	-	-	conv	input	-	-

image $i_t - 1, i_t, i_t + 1 = 3 + 3 + 3 = 9$

Layer type	input	size	strike	out	reg.	acti
concat	I_{t-1}	I_t	I_{t+1}	-	-	-
conv	32	3x3	(1,1)	32	BN	relu
conv	32	3x3	(1,1)	32	BN	relu
conv	32	3x3	(1,1)	32	BN	relu
sum2	-	-	conv	concat	-	-
concat	-	-	sum1	sum2	-	-

Encoder

1st intermediate after concatenating pixel vol and image = 64+32 = 96 – input

Layer type	input	size	strike	out	reg	acti
conv11	96	5x5	(1,1)	64	-	-
conv21	64	5x5	(1,1)	64	BN	relu
conv22	64	5x5	(1,1)	64	BN	relu
conv23	64	5x5	(1,1)	64	-	-

2nd intermediate - input 64-¿128 output(64*2=128)

Layer type	input	size	strike	out	reg	acti
conv31	64	5x5	(2,2)	64	BN	relu
conv32	64	5x5	(1,1)	64	BN	relu
conv33	64	5x5	(1,1)	64	BN	relu

RedBlocksx24

ResBlocks – 24 X 128 input -¿ 128 output

res channels = 128

Layer type	input	size	strike	out	reg.	acti
conv	128	5x5	(1,1)	128	BN	relu
conv	128	5x5	(1,1)	128	BN	relu
conv	128	5x5	(1,1)	128	BN	relu

Decoder

Layer type	input	size	strike	out	reg.	acti
conv41	128	3x3	(2,2)	64	-	-
add	-	-	conv41	conv23	-	-
dconv1	64	3x3	(1,1)	64	-	-
conv51	64	3x3	(1,1)	64	BN	relu
conv52	64	3x3	(1,1)	64	BN	relu

Layer type	input	size	strike	out	reg.	acti
dconv2	64	4x4	(2,2)	64	-	-
conv61	64	3x3	(1,1)	64	BN	relu
conv62	64	3x3	(1,1)	64	BN	relu
add	-	-	conv62	input	-	-

Project modifications: In fig.5, Video Deblurring Framework (as described above) has 24 residual blocks with 128 channels. Our larger model (Ours-large w/ CA) clearly outperforms EDVR both in PSNR and SSIM.

Video Deblurring Framework architecture deblur uses symmetric skip-connections between the encoder and decoder to reconstruct a resulting image from encoded features while preserving image structures (described above). There are 24

residual blocks between the encoder and decoder to effectively refine blurry feature maps in low-resolution. We apply ReLU and batchnorm for regularization after all convolution and deconvolution layers except the layers that are connected by skip-connections. Networks takes 4 inputs, (the previous, current, and next input frames, and pixel volume of the deblurred previous frame). The inputs into two groups, first to concatenated consecutive frames, and pixel volume. We feed each group into the feature transform layers and fuse their features by concatenation. To reduce complexity, Grayscale pixel are used for pixel volume. Color pixel will be computationally more expensive. Network's long skip-connection from the input frame to the network output helps the network to predict only residuals but preserves the overall contents of the input frame.

-

IV. *Literature Review*

Deep learning-based approaches. Several approaches use neural networks that are trained to automatically aggregate information from neighboring frames and reconstruct deblurred frames. Su et al. [2017] proposed a CNN that receives multiple frames concatenated together as input. They also showed rough motion compen- sation between input frames helps to deblur difficult scenes with large motions. Kim et al. [2017] and Nah et al. [2019] presented recurrent CNNs for video deblurring, and showed that utilizing previous results can improve deblurring quality. However, these methods do not work well for videos with large motion, as they do not use motion compensation. Recently, a spatio-temporal trans- former network [Kim et al. 2018] was proposed to improve motion compensation performance for video restoration tasks, such as video superresolution and video deblurring. The network estimates optical flow from multiple neighboring frames together to effectively handle occlusions. More recently, Wang et al. [2019] proposed an alignment module based on deformable convolutions for video restoration tasks that allows more effective utilization of information from other video frames. Video deblurring based on deconvolution also requires estimation of spatially-varying blur kernels. Early approaches [Bar et al. 2007; Wulff and Black 2014] used segmentation maps to reduce the ill- posedness and to effectively process blur caused by moving objects. Later methods [Kim and Lee 2015; Ren et al. 2017] proposed video deblurring frameworks based on optical flow to model inter-frame motion. These methods alternatingly perform optical flow estima- tion and image deblurring. However, they still assume relatively simple blur models, so can fail when applied to real-world videos that include complex blur.

V. *Data Set*

To train both LitFlowNet and project model, Video Deblurring Framework, use dataset of synthetically blurred videos [Su et al. 2017], which was generated by capturing sharp videos at high frame rate, and averaging adjacent frames. The dataset consists of 71 pairs of a blurry video and its corresponding sharp video, which provide 6,708 pairs of 1280×720 blurred and sharp frames in total. The videos in the dataset have blur caused by camera shake and object motion. Experiment will use 61 videos in the dataset as our training set, and 10 videos for test. The training and test sets consist of 5,708 and 1,000 pairs of images, respectively. For validation, 10 percent of the dataset will be used for validation.

VI. *Experiments*

In the experiment, first liteflownet is trained and then the Video Deblurring Framework. LiflowNet is trained with pretrained weight provided by author. For fine-tuning, we randomly selected two consecutive frames from the training video set. We then randomly cropped the same areas of size 256×256 from the selected frames, and fed them to the network. For each sequence, we iterate from the first to last cropped frame. For each iteration, we set the current cropped frame as the target blurry frame I_t^b and compute the gradient using the result of the previous frame $({}_{t-1}^{est}$ as well as the current target blurry frame and its neighboring blurry frames. Then, we update the network parameters by using backpropagation. The backpropagation is performed only for the current iteration, i.e., the gradient does not propagate to the network at the previous state through I_{t-1}^{est}. For the initial frame of each sequence, we set $I_{t-1}^{est} = I_t^b$ and $I_{t-1}^b = I_t^b$. We used normalized pixel values from -1 to 1, and set the batch size to eight. We used Adam optimizer [Kingma and Ba 2015] with $\beta 1 = 0.9$ and $\beta 2 = 0.999$. We initially set the learning rate to 0.0001, and decayed it with a decaying rate 0.1 for every 100,000 iterations. The network converged after 200,000 iterations with the learning rate of 0.000025. To train Video Deblurring Framework, we use mean absolute difference between deblurred frames and their corresponding ground truth sharp frames as the loss function. Although the network does not have any specific constraints or loss functions for temporal coherence, it generally produces temporally smooth results as our recurrent network utilizes the deblurred previous frame to process the current frame.

Model	PSNR	SSIM
litefolnet[9]	27.64	.894
Our Model	30.89	.891

The performance is measured in terms of peak signal-to-noise ratio $(PSNR)$ $[K\ddot{o}hler\ et\ al.\ 2012]$ whis is 30.89 and structural similarity index $(SSIM)$ $[Wang\ et\ al.\ 2004]$ which is. .891. The dataset is small for this test.

As per the documented performance measure of the PVDNet[3] with pixel-volume, the PSNR and SSIM are 31.63 and .91 respectively on the entire dataset.

The performance of our model, Video Deblurring Framework, is lower than PVDNet[3]. It is because of better

hyper-parameter tuning and the architecture also need some modifications.

Model Results

Fig. 7. left most image is the input image. The shake is visible in the brick smoke stack. The middle image is the model output (bo blur) and right most image is the ground truth.

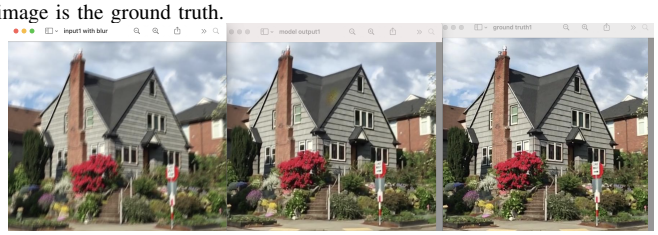


Fig. 8. left image is the blurred input cyclist image. The shake is visible in the cyclist legs. The right image is deblurred and legs look sharper.

Fig. 9. left image is the blurred input image. The shake is visible in the trees. The right image is deblurred and trees look sharper.

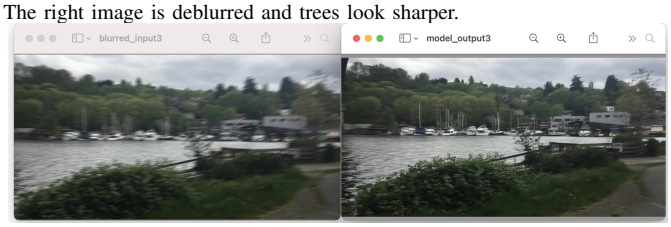


Fig. 10. left image is the blurred input image. The shake is visible in the letters specially "Hardware". The right image is deblurred and "Hardware" looks sharper.

A. Experiment details

VII. *Ablation Study*

To show the effectiveness of the liteFlowNet[9] and pixel volume[3] in our video deblurring framework, we performed an ablation study. We tested 2 models in this study. Our video deblurring framework that receives consecutive blurry frames as well as a deblurred previous frame. Then we try LiteFlowNet[9] which uses wrapping mechanism. Our video deblurring framework uses pixel volume and we can see the images are much sharper. Whereas liteFlowNet[9] uses wrapping mechanism, we can see the sharpness is not as good as our video deblurring framework.

Fig. 11. blurred 3 consecutive frames

Fig. 12. liteFlowNet model output

Fig. 13. Video Deblurring Framework model output

B. Evaluation methods

Deblurred $I_t^{s'}$ frame is obtained by wrapping I_{t-1}^{s} frame with the optical flow. To measure difference between the $I_t^{s'}$ and the ground truth I_t^s in terms of peak signal-to-noise ratio $(PSNR)$ [*Köhler et al.* 2012] and structural similarity index $(SSIM)$ [*Wang et al.* 2004]. We assume that a test frame with a low PSNR is likely to have large blur, because blur is the primary factor of structural differences from its ground truth.

VIII. *Quantitative Results*

The model deblur produced reasonable result. More experiments with Network parameter tuning would have helped better results.

IX. *Qualitative Findings*

Ref figure 10, the debur input video frame restored small-scale structures relatively well, but its results still have remaining blur and noisy artifacts due to large motions

between input frames that are not handled completed by model's motion compensation mechanism., "TrueValue" is still not fully deblurred. Also some of the letters in "Hardware" still has noise to be cleared. This suggests, the network needs more finetuning and some modifications are required in the network architecture.

X. *Challenges*

I have trained the entire network with 2 GPUs. Each training took 3-4 days. Hence, I couldn't get many parameter tuning opportunity. Also, The scope for network modifications depending on the results was limited. More examples with better GPU environment would hv yielded better inference.

XI. *Conclusion*

Our framework achieved the state-of-the-art deblurring performance, as demonstrated by experiments on videos with large blur. Combination of blur-invariant motion estimation learning and a pixel volume can improve motion compensation without significant computational overhead or changes in network design.

XII. *Team Member Contribution*

I am the only contributor for this project. The images in the project are taken from PVDNet[3] and liteFlowNet[2] .

XIII. *Github*

Text2Image

https://github.com/suvasis/suvasis or
https://github.com/suvasis/cs231a

REFERENCES

[1] https://files.is.tue.mpg.de/black/papers/sunIJCVflow2013.pdf
A Quantitative Analysis of Current Practices in Optical Flow Estimation and the Principles behind Them

[2] https://arxiv.org/pdf/1709.02371.pdf
PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume

[3] https://arxiv.org/pdf/2108.09982.pdf
Recurrent Video Deblurring with Blur-Invariant Motion Estimation and Pixel Volumes

[4] https://lmb.informatik.uni-freiburg.de/resources/datasets/
Learning Transferable Visual Models From Natural Language Supervision

[5] https://aclanthology.org/P18-1238.pdf
Middlebury datasets

[6] https://arxiv.org/pdf/2102.08981.pdf
The KITTI dataset was produced in 2012

[7] https://arxiv.org/pdf/1612.01925.pdf
FlowNet 2

[8] https://ps.is.mpg.de/research_projects/mpi-sintel-flow
MPI Sintel Flow Dataset

[9] https://arxiv.org/pdf/1805.07036.pdf
LiteFlowNet

[10] https://arxiv.org/pdf/2007.09319.pdf
LiteFlowNet3

[11] https://github.com/twhui/LiteFlowNet3
LiteFlowNet3 original implementation

[12] Khler\et\al.\2012
Rolf Köhler, Michael Hirsch, Betty Mohler, Bernhard Schölkopf, and Stefan" Harmeling. 2012. Recording and Playback of Camera Shake: Benchmarking Blind Deconvolution with a Real-World Database. In Proc. ECCV

[13] Wang\et\al.\2004
Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing 13, 4 (2004), 600–612.

[14] https://arxiv.org/pdf/1505.04597.pdf
U-Net