Photosequencing of Motion Blur using Short and Long Exposures
Supplementary PDF

Vijay Rengarajan[1*], Shuo Zhao[1], Ruiwen Zhen[2], John Glotzbach[2], Hamid Sheikh[2], Aswin C. Sankaranarayanan[1]
[1]Carnegie Mellon University, [2]Samsung Research America
[*]vangarai@andrew.cmu.edu

In this supplementary PDF, we provide the details of our neural network architecture, and comparison between inference approaches based on single and three trained models. The accompanying video provides all our results.

S1. Comparison of Single and Three Model Approaches

As mentioned in Sec. 4.2 of the main paper, since our technique uses recursive decomposition, the inputs to the network beyond the first level would have at least one noise-free estimated image. However, our training involves noise for both the short exposure images. We showed results using this single-model approach in the main paper. As a variation to our single trained model, we also trained three different models (with the same architecture) for the three cases with two, one, and zero noisy images for the short-exposure inputs. This is explained in Fig. S1.

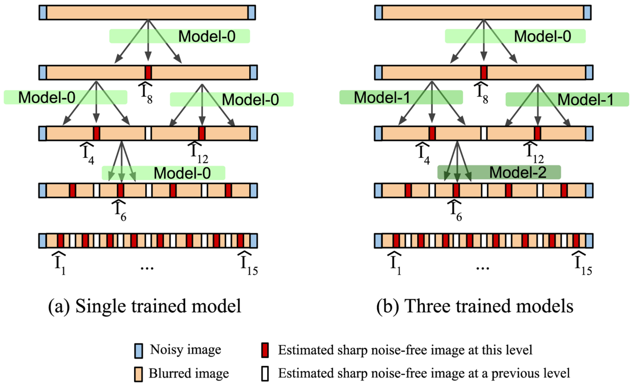
We observed no significant difference in our test results for these two approaches. This can be seen in Fig. S3. For four levels of decomposition as depicted in Fig. S1, we show the estimated images $\widehat{I}_4$, $\widehat{I}_{12}$ inferred using Model-1 and $\widehat{I}_6$ using Model-2 in Fig. S3(b) for *Skate* and *Jellyfish* scenes. The corresponding estimated images using the single trained model approach inferred using Model-0 is shown in Fig. S3(a). We can see no observable difference. The relative PSNRs between the estimated images of the two approaches are on the high end showing that both the approaches performs very similarly, and hence, we used the single trained model approach for all our results in the main paper.

S2. Network Architecture

The architecture of our neural network with layer numbering is shown in Fig. S2. The details of each layer, viz. input/output channels, filter size, padding, and stride, are provided in Table S1.

References

[1] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016. 2



(a) Single trained model (b) Three trained models

Figure S1. *Single and three trained model approaches.* (a) In the first approach, a single model is trained with two short exposure noisy images, and the same model is used at inference for all levels of decomposition even if one or both of the short exposure inputs are noise-free estimated images from a previous level. (b) In the second approach three models are trained with two, one, and zero noisy images as inputs. The respective model is used based on the number of estimated noise-free images involved for that particular inference set of inputs. For instance, the estimations of I_4 and I_{12} in (b) use Model-1 since it involves the estimated noise-free I_8 as one of its inputs and the other short-exposure input is noisy, and the estimation of I_6 uses Model-2 since both its short-exposure inputs I_4 and I_8 are noise-free.

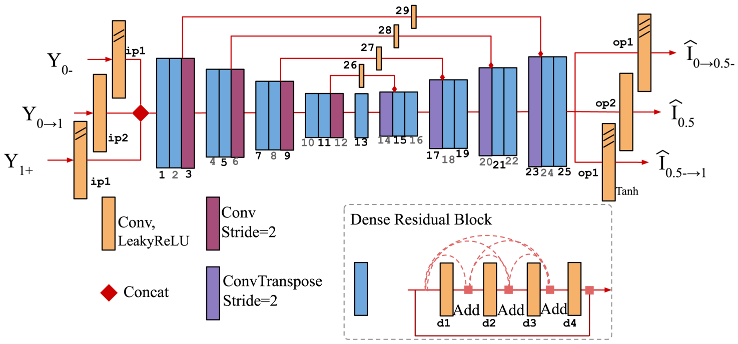
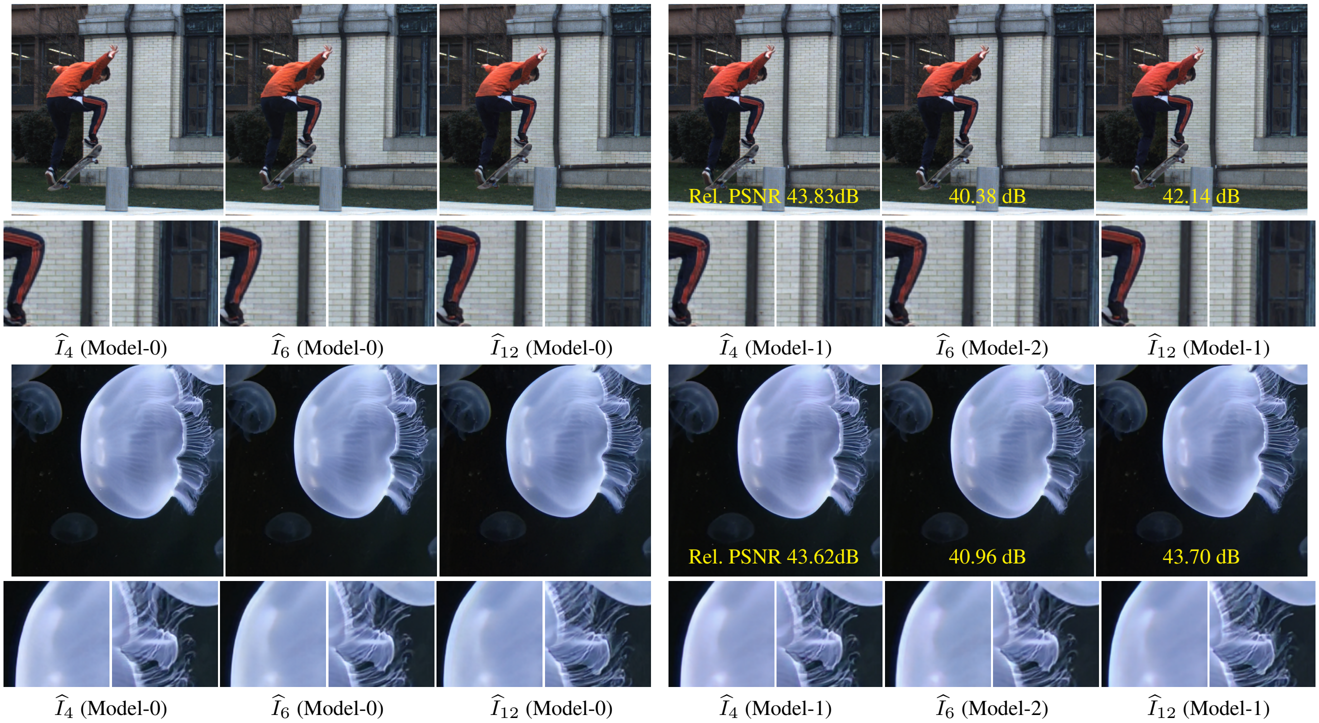


Figure S2. Network architecture.


Rel. PSNR 43.83dB
40.38 dB
42.14 dB
$\widehat{I}_4$ (Model-0)
$\widehat{I}_6$ (Model-0)
$\widehat{I}_{12}$ (Model-0)
$\widehat{I}_4$ (Model-1)
$\widehat{I}_6$ (Model-2)
$\widehat{I}_{12}$ (Model-1)
Rel. PSNR 43.62dB
40.96 dB
43.70 dB
$\widehat{I}_4$ (Model-0)
$\widehat{I}_6$ (Model-0)
$\widehat{I}_{12}$ (Model-0)
$\widehat{I}_4$ (Model-1)
$\widehat{I}_6$ (Model-2)
$\widehat{I}_{12}$ (Model-1)

(a) Single trained model

(b) Three trained models

Figure S3. *Comparison of single and three trained models approaches.* Both the approaches perform very similarly as shown in (a) and (b) for the estimation of three images of the sequence depicted in Fig. S1. The relative PSNRs between the two approaches are quite high denoting this behavior.

Table S1. Layer details of our architecture in Fig. S2

Layer	Type	chan-in → chan-out	filt, pad, stride
ip1	Conv	3 → 16	7x7, 3, 1
ip2	Conv	3 → 32	7x7, 3, 1
1	ResB	64 → 64	3x3, 1, 1
2	ResB	64 → 64	3x3, 1, 1
3	Conv	64 → 128	3x3, 1, 2
4	ResB	128 → 256	3x3, 1, 1
5	ResB	128 → 256	3x3, 1, 1
6	Conv	128 → 256	3x3, 1, 2
7	ResB	256 → 512	3x3, 1, 1
8	ResB	256 → 512	3x3, 1, 1
9	Conv	256 → 512	3x3, 1, 2
10	ResB	512 → 512	3x3, 1, 1
11	ResB	512 → 512	3x3, 1, 1
12	Conv	512 → 1024	3x3, 1, 2
13	ResB	1024 → 1024	3x3, 1, 1

Layer	Type	chan-in → chan-out	filt, pad, stride
14	ConvT	1024 → 256	4x4, 1, 2
15	ResB	512 → 512	3x3, 1, 1
16	ResB	512 → 512	3x3, 1, 1
17	ConvT	512 → 128	4x4, 1, 2
18	ResB	256 → 256	3x3, 1, 1
19	ResB	256 → 256	3x3, 1, 1
20	ConvT	256 → 64	4x4, 1, 2
21	ResB	128 → 128	3x3, 1, 1
22	ResB	128 → 128	3x3, 1, 1
23	ConvT	128 → 32	4x4, 1, 2
24	ResB	64 → 64	3x3, 1, 1
25	ResB	64 → 64	3x3, 1, 1
op1	Conv	64 → 3	3x3, 1, 1
op2	Conv	64 → 3	3x3, 1, 1
26	Conv	512 → 256	3x3, 1, 1
27	Conv	256 → 128	3x3, 1, 1
28	Conv	128 → 64	3x3, 1, 1
29	Conv	64 → 32	3x3, 1, 1

Conv - convolutional layer, ConvT - transpose convolutional layer [1].
All convolutional layers are followed by Leaky ReLU. Only op1 and op2 are followed by (tanh()+1)/2.

Structure of Dense Residual Block (ResB)

Layer	Type	chan-in → chan-out	filt, pad, stride
d1	Conv	nchan → nchan	3x3, 1, 1
d2	Conv	nchan → nchan	3x3, 1, 1
d3	Conv	nchan → nchan	3x3, 1, 1
d4	Conv	nchan → nchan	3x3, 1, 1