Learning Spatially Varying Pixel Exposures for Motion Deblurring

Cindy M. Nguyen, Julien N. P. Martel, and Gordon Wetzstein

Abstract—Computationally removing the motion blur introduced by camera shake or object motion in a captured image remains a challenging task in computational photography. Deblurring methods are often limited by the fixed global exposure time of the image capture process. The post-processing algorithm either must deblur a longer exposure that contains relatively little noise or denoise a short exposure that intentionally removes the opportunity for blur at the cost of increased noise. We present a novel approach of leveraging spatially varying pixel exposures for motion deblurring using next-generation focal-plane sensor–processors along with an end-to-end design of these exposures and a machine learning–based motion-deblurring framework. We demonstrate in simulation and a physical prototype that learned spatially varying pixel exposures (L-SVPE) can successfully deblur scenes while recovering high frequency detail. Our work illustrates the promising role that focal-plane sensor–processors can play in the future of computational imaging.

Index Terms—Motion deblurring, programmable sensors, in-pixel intelligence, end–to-end optimization, computational imaging

1 Introduction

CAMERAS have become ubiquitous, their presence felt in every smartphone and countless other devices sold today. Whether their images are designed for social media or processed by a self-driving car, modern cameras are still susceptible to the age-old problem of motion blur. This artifact is the result of either object or scene motion during the image exposure or camera shake via handheld photography, thus rendering the picture of a touching memory or an image used for computer vision tasks useless.

Unfortunately, removing motion blur remains an arduous task. Due to the heterogeneity of local and global motion blur, blind deblurring is difficult to address with pure deconvolution. Recent deep learning methods have popularized multi-scale [1], [2], [3], [4] or multi-temporal [5], [6] deep learning approaches to address this issue. The coarse-to-fine nature of these networks allows for the gradual refining of motion deblurring kernels that are applied spatially invariantly, even in cases of non-uniform motion blur.

These methods have demonstrated the power of machine learning for deblurring, but notably do not take advantage of offloading computation to the hardware. On the other hand, computational imaging methods have excelled in combining software and hardware solutions to simplify many ill-posed computer vision tasks. These include engineering point spread functions (PSFs) [7], [8], [9] and custom coded exposures [10], [11], [12], [13], [14] for motion deblurring. However, these methods are often limited to heuristic designs, and often restricted by fabrication limits, sensor capabilities, or human intuition.

Fortunately, we are on the brink of a new era of sensor design. The rise of programmable sensors [15], [16], [17], in which sensing and processing can be executed together on the same silicon chip, has brought forth a new age of computing that operates at the time of image capture. Programmable sensors, or focal-plane sensor–processors, open up avenues for analog, digital, or mixed signal processing directly in-pixel. These sensors provide opportunities to preserve optical information of a scene that is otherwise irreversibly destroyed during the sensor integration or capture process. Moreover, the in-pixel intelligence running on these sensors can be learned using end-to-end (E2E) design strategies that jointly optimize the in-pixel programs with downstream computer vision algorithms.

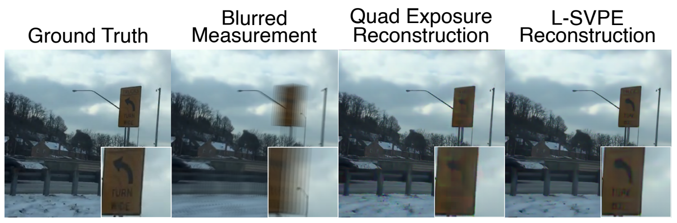


Fig. 1. An example reconstruction. The ground truth is the first frame of an input video segment, and the blurred image is all eight frames of the video segment averaged, which is the same as a 30 fps capture. The reconstruction from the *Quad* exposure [14] with bilinear interpolation shows more motion and color artifacts compared to that of the learned spatially varying pixel exposure (*L-SVPE*).

Given these capabilities, we address the above concerns with our jointly Learned Spatially Varying Pixel Exposures (L-SVPE) and machine learning–based reconstruction network to perform high quality motion deblurring (Fig. 1). The benefits are two-fold. Within short exposures, the low signal-to-noise ratios are challenging to handle, but these short exposures still retain high frequency information that we desire in a reconstruction. Within longer exposures, images will have reduced noise levels thanks to signal accumulation. However, as the sensor integration time increases, these longer exposures are more prone to motion blur. We

• *C. M. Nguyen, J. N. P. Martel, and G. Wetzstein are with the Department of Electrical Engineering, Stanford University.*

seek to use the complementary information from both in a single snapshot in our E2E framework.

As programmable sensors grow in production and popularity, we believe in the need to develop appropriate computational imaging algorithms to take advantage of these new capture capabilities. Future mobile phones may no longer be limited to the fixed global exposures they have today for capturing scenes with dynamic motion or low lighting. High dynamic range (HDR) imaging, most popularly done with some form of exposure bracketing [18], will struggle less with image alignment since programmable sensors can capture information at multiple exposures in a single snapshot [19], [20]. We demonstrate that motion blur is just one of many tasks that highlight the utility of these powerful sensors.

Specifically, we make the following contributions:

- We introduce a fully differentiable model with a learned in-pixel encoder and deep convolutional decoder for motion deblurring. The encoder can be implemented on a programmable sensor offering "in-pixel intelligence."
- We demonstrate in simulation that the E2E optimization of the coded exposures along with the decoding network yields superior reconstructions of motion blurred images over those of non-optimized exposures.
- We implement a prototype camera using the SCAMP-5 programmable sensor–processor and demonstrate that our results translate to real-world captures, where the coded exposures are realized electronically.

Source code and trained models will be made available upon publication.

2 Related Work

Motion deblurring. Early examples of blind motion deblurring include learning motion blur kernels using CNNs [21], [22], [23]. More recent examples increase the receptive field by using multi-resolution inputs [1], [2], [3], [4] to learn on a combined global and local scale. Attention [3] and atrous convolutions [24], [25] can help apply spatially varying weights for local blurs. Inspired by transformers, Tu et al. [26] use multilayer perceptrons (MLPs) and attention with multi-resolution features for deblurring, reducing the required number of learnable parameters. Though these methods have produced plausible reconstructions, deblurring networks are fundamentally limited due to the ill-posed nature of the problem, which can only be overcome by changing the image formation model through methods like E2E optimization.

End-to-end optimization. Jointly designing optics and reconstruction networks has emerged as an incredibly useful paradigm in computational imaging [27]. This E2E optimization has been applied to a number problems such as extended depth of field [28], depth estimation [29], [30], [31], HDR [32], [33], super resolution localization microscopy [34], lensless imaging [35], [36], and Fourier ptychographic microscopy [37]. However, once the optics are fabricated, the challenge of calibrating them arises. To alleviate the need for calibration, in-pixel sensing strategies can also be designed in an E2E fashion for general capture [38], HDR [19], [20], and compressive imaging [39]. We propose following this route, learning a spatially varying exposure pattern jointly with a network for the task of motion deblurring.

Computational imaging for deblurring. Deblurring can be made less of an ill-posed problem with a variety of hardware modifications. Raskar et al. [10] use a random binary coded global exposure pattern, implemented via a liquid-crystal shutter. The coded exposure provides a PSF which preserves high spatial frequencies, allowing the blur to be decoded using traditional deconvolution algorithms. Agrawal and Xu [13] further improve this design with heuristics on PSF invertibility and estimation. Other works use a custom rolling shutter [11] or deblur using information from a camera's default rolling shutter [40]. Jeon et al. [41] use communication theory to design fluttering patterns, while more recently, Jiang et al. [14] use an interlaced short, medium, and long exposure pattern with a specialized network for reconstruction. We compare this exposure pattern with ours, while using a simpler network for reconstruction. Elmalem et al. [8] design an optic that encodes blur in its color PSF, and Yosef et al. [9] build on this work by additionally using a focus mechanism to recover video frames from a capture with motion blur. Rengarajan et al. [42] proposed using short-long-short exposures with recursive blur decomposition to deblur. Notably, all the above approaches have been designed using heuristics and theory, making the search space limited to human intuition.

Programmable sensors. The emergence of programmable sensors, offering unprecedented flexibility of in-pixel processing, has brought forth numerous interesting ideas for computational photography. These sensors, also known as focal-plane sensor–processors [16], conduct low-level image processing during the capture. They reduce the need for excessive computational post-processing and enable new capture processes of an image to preserve information that would be lost otherwise, such as dynamic range. Programmable sensors, such as the SCAMP-5 [15] which we use as our prototype, offer programmable pixels and have been successfully used in HDR imaging [19], video compressive imaging [19], [43], depth from defocus [44], feature classification [45], and ego-motion estimation [46]. More recently, coded exposures have also been used for compressive light-field and hyperspectral imaging [39]. Here, we propose to use one variant of these new vision chips, the SCAMP-5, to program pixel-wise coded exposures. To our knowledge, this is the first application of these sensor–processors to motion deblurring.

3 Formulation

We simulate the capture of a scene with our learned programmable sensor using spatially varying pixel exposures (Sec. 3.1), which we also refer to as coded exposures. We then form a multi-channel image with C channels from the single snapshot, where C represents the number of unique exposure lengths in the learned coded exposure. We do so by interpolating (Sec. 3.2) exposure pixels that were not explicitly captured. This image stack with a resolution of

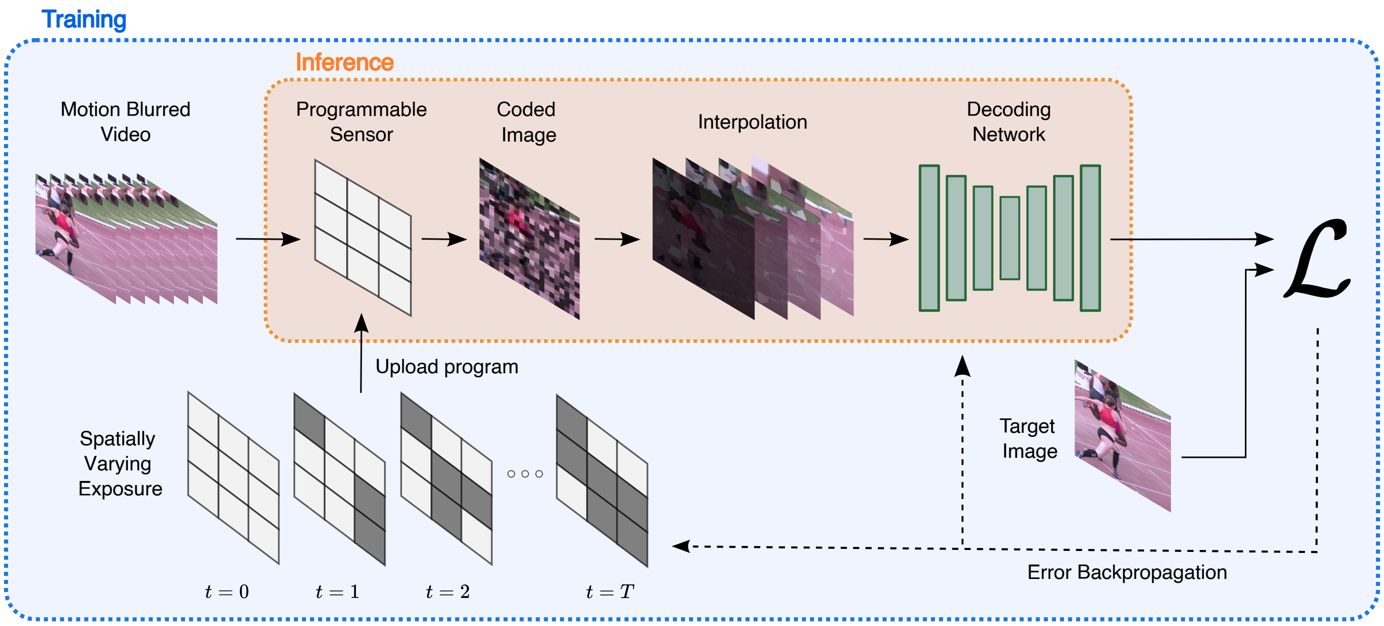


Fig. 2. Illustration of our *L-SVPE* optimization framework, including the learned coded shutter, interpolation step, and decoding network. The error between the reconstructed image and ground truth is backpropagated to the coded exposure lengths and decoding network. Here, T represents the maximum exposure length. This pipeline illustrates an example use of four different exposure lengths, providing four full-resolution interpolations.

$H \times W \times C$ pixels is fed as the input of the decoding network (Sec. 3.3) to produce a reconstructed image. The pipeline is illustrated in Fig. 2.

3.1 Learned spatially varying pixel exposures

We model the exposure at pixel location p as the integration of the incident irradiance V_p over an exposure time Δt, which can be written as

$$E_p(t) = \int_t^{t+\Delta t} V_p(t')dt'. \tag{1}$$

Here, we let E_p represent the exposure at pixel p. We introduce a learned spatially varying exposure time Δp for each pixel p to indicate an "on" (contrary to "off") shutter. Thus Eq. 1 can be rewritten as

$$E_p(t) = \int_t^{t+\Delta p} V_p(t')dt'. \tag{2}$$

The exposure E_p relates to the captured image via the camera response function as $I_p(t) = \mathcal{R}(E_p(t))$, which captures the noise and quantization effects of the camera.

3.2 Exposure length–specific interpolation

The single-shot measurement on our focal-plane sensor–processor is a single-channel, grayscale image. We expand the measurement as a preconditioning step before decoding. Since our learned exposure times will vary spatially, we interpolate the missing exposure pixels to generate a full resolution estimate of the utilized exposure lengths. In our framework, we discretize the continuous varying exposure time t to a set of discrete values $\mathcal{S} = \{1, .., T\}$, in which T is the maximum exposure time. We then interpolate all pixels of the single-channel sensor image with the same exposure to form a separate channel of the image stack that serves as the input to our motion deblurring network. We argue this interpolation step speeds up training since the learned kernels in the decoding network can be then applied spatially invariantly. We apply the interpolation function $\mathcal{U} : \mathbb{R}^{H \times W} \rightarrow \mathbb{R}^{H \times W \times C}$ where C is the unique number of exposure lengths used from $\mathcal{S}$.

We use two different types of interpolation: Bilinear and Scatter-weighted interpolation. Bilinear interpolation (denoted as **B**) is applied in cases where the exposures have a 2×2 (Quad) or 3×3 (Nonad) tiled arrangement (see Sec. 5.3), using information from regular sampling grids. Scatter-weighted interpolation (denoted as **S**) is applied in cases where pixel exposures of the same length have varying distances from each other across the sensor. Scatter interpolation is applicable when the coded exposures are random or learned, but this method can also be used in the tiled case. Scatter interpolation requires finding the N closest neighbors using a fast k-d tree [47] and adding the values of neighboring pixels together, each weighted by their inverse distance to the point of interest to some power r. The interpolated value at point p can be written as

$$\mathcal{U}(I_p) = \frac{\sum_{i=1}^{N} w_i(p) I_i}{\sum_{i=1}^{N} w_i(p)}, \tag{3}$$

where I_i is the value at neighboring pixel i. The weighting function $w_i(p) = \frac{1}{d(p,p_i)^r}$ is computed using Euclidean distance as d. Also, N represents the number of nearest neighbors sharing the same exposure length as pixel p.

3.3 Capture decoding

Once we obtain an interpolated multi-channel image, we decode this using the well-known U-Net architecture [48].

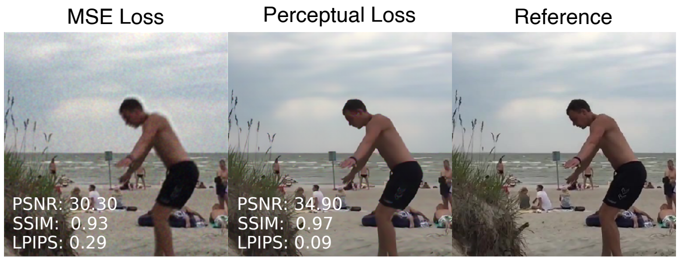


Fig. 3. Qualitative comparison between the reconstructions of two learned exposure models using *L-SVPE* trained with an L_2 loss and our perceptually-guided loss. As LPIPS decreases, the quality of the reconstruction increases. L_2 lends itself to more blurry reconstructions, which may miss high frequency details.

Our CNN $W_\psi : \mathbb{R}^{H\times W\times C} \rightarrow \mathbb{R}^{H\times W}$ contains skip connections with a depth of 6 without batch normalization, and each downsampling block contains a single convolution and ReLU with learned parameters ψ. The upsampling blocks use ConvTranspose to upsample the features at each stage. Our decoding step can be written as

$$\hat{Y_p} = W_\psi(\mathcal{U}(I_p)), \tag{4}$$

where $\hat{Y_p}$ is the reconstructed grayscale value at pixel p. We opt for the widely used U-Net architecture that has been useful for a variety of imaging problems [49], [50] and has less parameters than the state-of-the-art deep deblurring methods [1].

3.4 Loss

We use a linear combination of an MSE loss (L_2) and a perceptual loss using VGG features (L_{VGG}) [51]. Our loss can be written as

$$L_{\text{percep}} = L_2 + \lambda L_{\text{VGG}}, \tag{5}$$

where λ is the coefficient to weight the VGG-based loss. We use $\lambda = 100$.

We compute the loss between the reconstruction with the clean first frame of the video segment, which we use as ground truth. We choose the first frame to reduce the need to deblur and prioritize using the information from the shortest exposure, as denoising is typically easier than deblurring.

The VGG-based loss is exceedingly helpful in reaching the perception–distortion trade-off [52], [53], which describes networks as trading off PSNR and SSIM performance for perceptual quality. Since there are many plausible reconstructions for motion blurred images, we find that the perceptual loss helps find a more suitable reconstruction over an unregularized MSE loss, which optimizes for per-pixel accuracy (Fig. 3).

4 Implementation

Dataset. To train our network and evaluate its performance, we use the Need for Speed (NfS) dataset [54], which consists of 100 videos obtained from the internet. Each video is captured at 240 frames per second (fps) with a 1280×720 resolution that we center crop to 512×512. We allocate 80 videos for our training set and 20 videos for our test set. For each video, we select 8 random 8-frame-long segments within the video, and each segment is averaged per-pixel to simulate a 30 fps capture. This processing produces 640 video segments for training and 160 video segments for testing. We augment these segments during training by introducing random flips and rotations, and normalize the images to be within $[0, 1]$. Our input to the E2E model is an 8-frame video segment, and we use the first frame of the video segment, equivalent to a Short exposure, as the ground truth frame.

Training. Our model was implemented in PyTorch and trained using an NVIDIA Quadro RTX 6000 GPU with 24 GB. Our model was trained for 1000 epochs with the AdamW [55] optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$). For networks with learned coded exposures, we use an exposure learning rate of 2×10^{-4} and a decoder learning rate of 5×10^{-4}. For baselines with fixed exposures, we use a decoder learning rate of 2×10^{-4}.

Metrics. We use three metrics to quantify the quality of our results. The first two are PSNR and SSIM [56] as traditional image quality metrics. We also use the perceptual metric LPIPS [51]. We provide both quantitative and qualitative evaluation of our method compared to baselines (Sec. 5.3), and we demonstrate the utility of each component in our ablation studies (Sec. 5.4).

5 Experiments

We perform a series of experiments to highlight the value of our learned coded exposure. Specifically, we evaluate our method on different coded exposures and ablate several design choices made in our E2E model.

5.1 Baselines

To demonstrate the utility of our learned exposure, we compare its performance against the following fixed exposures, where **B** represents Bilinear interpolation and **S** represents Scatter interpolation. Our baselines (Fig. 4) include:

- **Burst Average**: All 8 frames are averaged together, simulating a Long exposure, and compared to the ground truth. This baseline does not use a decoder.
- **Short**: We simulate a 240 fps capture which is equivalent to a single frame of the video segment input.
- **Medium**: We simulate a 120 fps capture by averaging the first four frames of the input.
- **Long**: We simulate a 30 fps capture by averaging all eight frames of the input.
- **Uniform Random (S)**: We initialize a fixed 512×512 array of pixel exposures uniformly selected from length 1 through 8.
- **Poisson Random (S)**: We use a multi-class Poisson disk sampling algorithm [57] to generate a Poisson-distributed fixed pixel exposure from length 1 through 8.
- **Nonad (nine-tuple) (B, S)**: We use the full range of exposures in a 3×3 arrangement. We randomly arrange an array of pixel exposures, each pixel a unique length from 1 through 8, with an additional exposure length 1 to make a total of 9 pixels in the

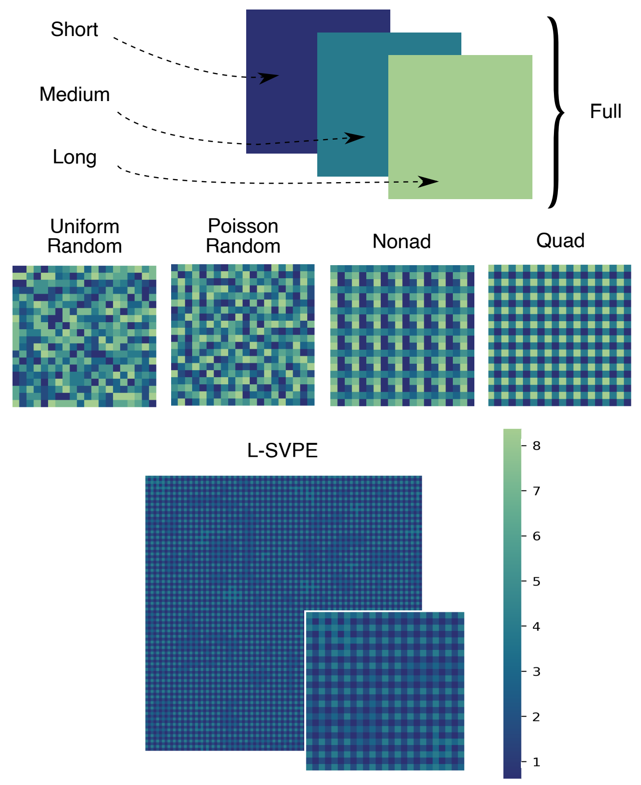


Fig. 4. Visualization of coded exposures of each baseline. *Short*, *Medium*, and *Long* are fixed global exposures. The *Full* consists of all three global exposures concatenated together. Each coded exposure (*Uniform Random*, *Poisson Random*, *Nonad*, and *Quad*) contains individual pixel exposures that can span from lengths 1 through 8, with cropped versions of the full resolution exposure shown here. We display the full resolution and a crop of the *L-SVPE* exposure to highlight the differences from *Quad*. Note that the average pixel exposure of *L-SVPE* is lower than that of *Quad*.

3×3. We then tile this fixed arrangement to fill the 512×512 resolution.

- **Quad (B, S)**: Following Jiang et al. [14], we use a fixed tile coded arrangement of LMMS (long-medium-medium-short), where the short pixel is 240 fps, medium pixels are 120 fps, and long pixel is 30 fps. We use this baseline to also initialize L-SVPE which explains its resemblance (Fig. 4).
- **Full**: We concatenated a full-resolution stack of the Short, Medium, and Long exposure all captured with the same start time and same viewpoint. This baseline serves as theoretical upper bound for how well our method can do without requiring an interpolation step and with full resolution information at different exposure times. This stack of captures would be physically impossible to capture but can be easily simulated.

For fair comparison, we use the same decoder network as our method on all baselines except the Burst Average, which does not use a decoding network. L-SVPE, by default, uses Scatter interpolation. See the supplemental material for more details on our choice of baselines.

TABLE 1
Comparison of interpolation methods. We show the efficiency and accuracy of Bilinear (denoted with B) and Scatter (denoted with S) interpolation methods with varying parameters. Time denotes the time required to interpolate a single image on average. Bilinear interpolation is the fastest for both *Quad* and *Nonad*. Scatter interpolation takes noticeably longer with the need to search for neighbors and is less accurate due to information dilution from further pixels. All Scatter-based methods here use $r = 1$.

Exposure	k Neighbors	Time (s)	PSNR ↑	SSIM ↑	LPIPS ↓
Quad [14] (B)	—	**0.001**	37.257	0.975	**0.049**
Quad (S)	3	0.041	42.138	**0.988**	0.062
Quad (S)	4	0.042	**42.826**	**0.988**	0.063
Quad (S)	5	0.044	41.914	0.986	0.076
Nonad (B)	—	**0.001**	30.733	0.837	0.089
Nonad (S)	3	0.117	28.912	0.826	0.127
Nonad (S)	4	0.124	28.783	0.825	0.150
Nonad (S)	5	0.131	28.520	0.822	0.154

5.2 Determining interpolation parameters

To determine the optimal parameters for Scatter interpolation, we compute the time it takes to perform interpolation with k neighbors. We compute the average time and accuracy of interpolating a test set of video segments captured with the chosen exposure. We first simulate the exposure capture of each video segment using the spatially varying exposures under the Exposure column to get a single-channel capture. We then use Bilinear or Scatter interpolation to interpolate pixels of exposure lengths that were not explicitly captured to acquire a $H \times W \times C$ multi-channel image. The Time column shows the time required in seconds to compute this step. We then compute the accuracy of each interpolation of each individual channel and average the metrics for each channel together for each image. We then compute the average of the metrics over the entire test dataset.

We present these results in Table 1. We test interpolation on the Quad and Nonad exposures. Bilinear interpolation (denoted as B) does not have a k parameter. We observe that as k increases, so does the time needed for computation per image. Scatter interpolation (denoted as S) is slower than Bilinear, but in the case of Quad, can provide more accurate interpolations. With Nonad exposure, we observe that Bilinear performs best, while Scatter degrades the quality of the reconstruction since neighbors are much further in the Nonad case. Thus, for all our Scatter-based methods, we use $N = 3$ and $r = 1$ which allows for relatively fast computation and accuracy.

5.3 Comparison against baselines

Figure 5 presents qualitative comparisons between these baselines. The Short exposure performs better than Medium and Long exposures, due to the network more easily learning to denoise than deblurring. The Full exposure outperforms all single exposure baselines, demonstrating the utility of combining information from different exposure lengths.

We see that baselines with spatially variant pixels outperform the Medium and Long baselines, while also notably performing better than the Short exposure signficantly in

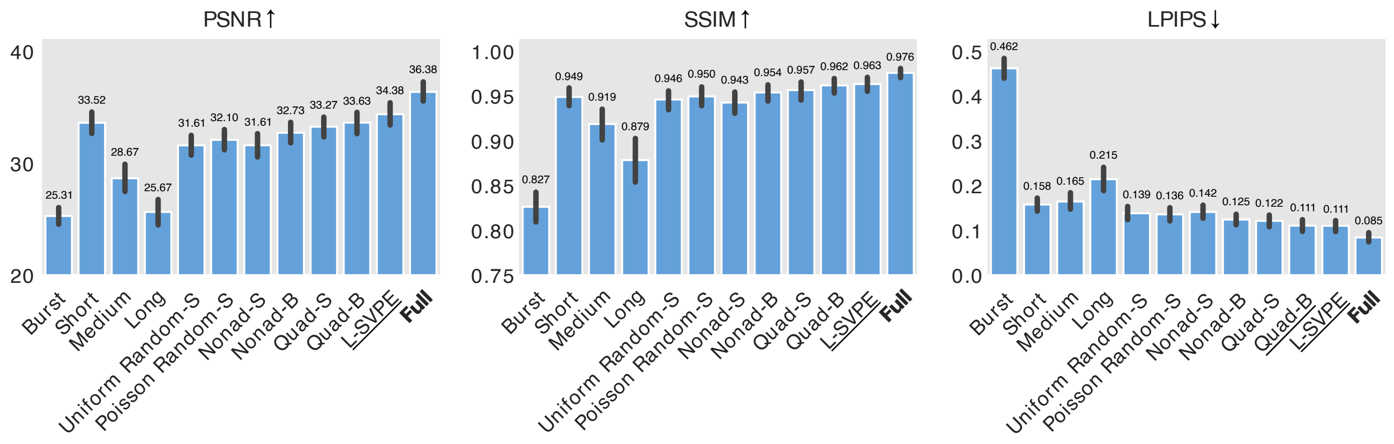


Fig. 5. Quantitative comparison of baselines. Printed values denote the numerical average, while gray bars represent the standard deviation. The *Full* exposure represents a theoretical upper bound to our method, and *L-SVPE* outperforms better than any of the fixed baselines, most closely reaching the theoretical upper bound on all metrics. These results are computed using the NfS dataset described in the text.

LPIPS. This result demonstrates the utility of varying exposures for perceptual quality. More structured shutters, like Nonad and Quad, do better than random shutters likely because of the uniformity of data given from the sensor. Our approach with learned exposures can outperform all baselines, reaching the closest to the theoretical upper bound, Full, in performance.

5.4 Ablation studies

Table 2 presents ablation studies on each component of the network. We compare our decoder choice, U-Net, against DnCNN [58], another popular memory-efficient image reconstruction network. The DnCNN is trained to predict the residual noise of the first channel of the multi-channel input into the decoder, if applicable. These studies demonstrate the utility of each component of our method. Note that the Scatter interpolation may not provide the best PSNR performance over no interpolation, but improves SSIM and LPIPS, which is consistent with our baselines study results. We found that DnCNN with a perceptual loss can be often detrimental to performance, which may suggest that DnCNN is not universally optimal for providing perceptually plausible deblurring from spatially varying exposures.

TABLE 2
Ablation studies showing the effect of the choice of shutter, interpolation, network, and loss on performance. We use (S) to denote methods which use Scatter interpolation. No (S) denotes that no interpolation step was used.

Exposure	Network	Loss	PSNR↑	SSIM↑	LPIPS↓
Uniform Random	DnCNN [58]	L_2	30.345	0.913	0.338
Uniform Random	DnCNN	Percep.	12.269	0.272	0.763
Uniform Random (S)	DnCNN	L_2	30.325	0.920	0.319
Uniform Random (S)	DnCNN	Percep.	28.255	0.915	0.225
Uniform Random	U-Net [48]	L_2	33.149	0.950	0.181
Uniform Random	U-Net	Percep.	28.877	0.905	0.200
Quad [14]	DnCNN	L_2	31.435	0.918	0.355
Quad	DnCNN	Percep.	11.719	0.247	0.752
Quad	U-Net	Percep.	32.345	0.952	0.140
L-SVPE	DnCNN	L_2	30.025	0.860	0.369
L-SVPE	DnCNN	Percep.	12.434	0.268	0.751
L-SVPE (S)	DnCNN	Percep.	29.530	0.919	0.230
L-SVPE	U-Net	L_2	27.061	0.865	0.406
L-SVPE	U-Net	Percep.	**34.625**	<u>0.961</u>	<u>0.122</u>
L-SVPE (S)	U-Net	L_2	28.987	0.892	0.320
L-SVPE (S)	U-Net	Percep.	<u>34.522</u>	**0.967**	**0.105**

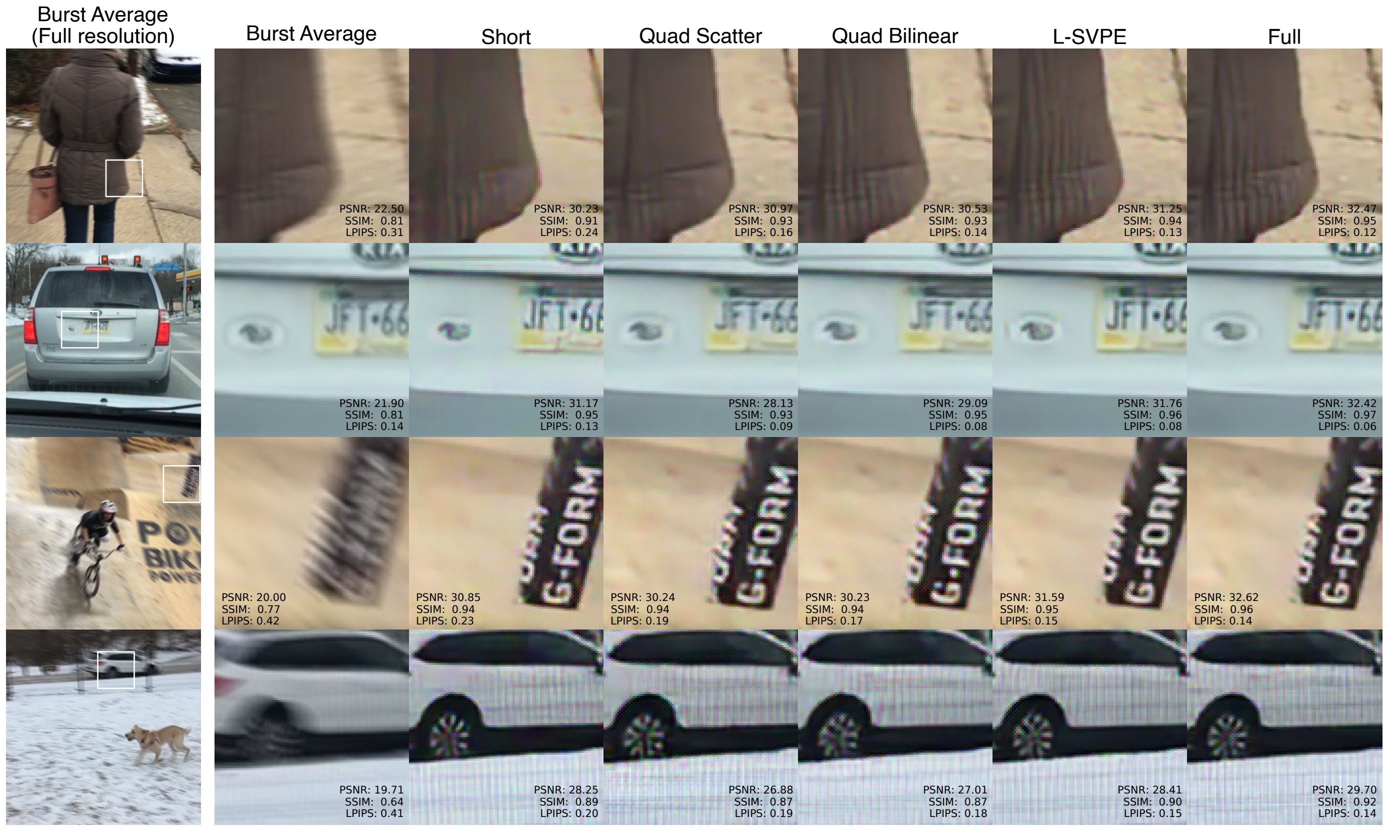
5.5 Image reconstruction quality

In Figure 6, we present a few qualitative examples of reconstructions with the top-performing baselines. To produce reconstructed RGB images, we individually reconstruct each channel with our baselines, which are trained on grayscale images. L-SVPE generalizes well across different color channels, while methods such as Short and Quad Bilinear can introduce discolored artifacts. We highlight the robust performance of our model against optical blur observed in Row 2 and global blur observed in Row 3. Row 1 and Row 4 demonstrate how L-SVPE is capable of recovering high frequency detail in the wrinkles of the coat and the fencing in front of the white car, respectively.

6 Prototype

We implement a physical prototype of our model using a focal-plane sensor–processor. Specifically, we use the SCAMP-5 [15], a 256×256 pixel array in which each pixel contains a programmable processing element (PE). Each PE contains a small number of analog and single-bit digital memories that can be set using dedicated instructions. We program the learned coded exposure pixel-wise into the analog memories using a micro-controller.

Figure 7 shows reconstructions from trained global exposure models (Long, Medium, and Short) and the two top-performing spatially varying exposures (Quad with Bilinear interpolation and L-SVPE) on two captured scenes, *Swings* and *Jump Rope*. Although Long and Medium reconstructions preserve the static background details, there are few improvements in motion deblurring for both scenes. The Short model tends to blur high frequency details such as the grass in *Swings* and the shoe detail in *Jump Rope* in an attempt to denoise the image. Quad Bilinear introduces blurring in areas with fine detail, losing details in the overhead wires in *Swings* and the white edges of the tile in *Jump Rope*. Only L-SVPE can successfully recover the fine details of each scene



Burst Average (Full resolution)
Burst Average
Short
Quad Scatter
Quad Bilinear
L-SVPE
Full
PSNR: 22.50 SSIM: 0.81 LPIPS: 0.31
PSNR: 30.23 SSIM: 0.91 LPIPS: 0.24
PSNR: 30.97 SSIM: 0.93 LPIPS: 0.16
PSNR: 30.53 SSIM: 0.93 LPIPS: 0.14
PSNR: 31.25 SSIM: 0.94 LPIPS: 0.13
PSNR: 32.47 SSIM: 0.95 LPIPS: 0.12
PSNR: 21.90 SSIM: 0.81 LPIPS: 0.14
PSNR: 31.17 SSIM: 0.95 LPIPS: 0.13
PSNR: 28.13 SSIM: 0.93 LPIPS: 0.09
PSNR: 29.09 SSIM: 0.95 LPIPS: 0.08
PSNR: 31.76 SSIM: 0.96 LPIPS: 0.08
PSNR: 32.42 SSIM: 0.97 LPIPS: 0.06
PSNR: 20.00 SSIM: 0.77 LPIPS: 0.42
PSNR: 30.85 SSIM: 0.94 LPIPS: 0.23
PSNR: 30.24 SSIM: 0.94 LPIPS: 0.19
PSNR: 30.23 SSIM: 0.94 LPIPS: 0.17
PSNR: 31.59 SSIM: 0.95 LPIPS: 0.15
PSNR: 32.62 SSIM: 0.96 LPIPS: 0.14
PSNR: 19.71 SSIM: 0.64 LPIPS: 0.41
PSNR: 28.25 SSIM: 0.89 LPIPS: 0.20
PSNR: 26.88 SSIM: 0.87 LPIPS: 0.19
PSNR: 27.01 SSIM: 0.87 LPIPS: 0.18
PSNR: 28.41 SSIM: 0.90 LPIPS: 0.15
PSNR: 29.70 SSIM: 0.92 LPIPS: 0.14

Fig. 6. Qualitative comparison of the five top-performing baselines. Each row shows an example frame of the full resolution average of all the video segment frames (*Burst Average*), a 100×100 crop of a region of the *Burst Average*, and the same crop of the reconstructions from the top five performing baselines. Reconstructions of all baselines are available in the supplemental material. **Row 1:** The high frequency detail of the jacket is well recovered by *L-SVPE* while blurred out by other methods. **Row 2:** The *Short* baseline fails to capture the border of the bumper sticker while introducing colored artifacts. *L-SVPE* correctly recovers the border and provides sufficient clarity to the lettering of the license plate. **Row 3:** The coloring artifacts can be observed in the *Short* and *Quad*-based baselines. **Row 4:** Competing baselines fail to recover the straight fencing in front of the white car fully, while *L-SVPE* is able to reconstruct the fencing covering the entire car.

in addition to deblurring. More details on how the captures were processed and videos of these scenes can be found in the supplemental material.

7 Discussion

We present a novel method for motion deblurring using learned coded pixel exposures. We demonstrate that our joint hardware-software approach is better than deep learning for comparable reconstruction networks. L-SVPE is able to address dynamic motion blurs and varying levels of noise across many different scenes. We demonstrate that its performance translates to a physical prototype, in which we show that L-SVPE can deblur while preserving high frequency details in real-world scenes.

Limitations Our programmable sensor operates in grayscale, without any color filter on top of the sensor. Thus, we design our method around capturing grayscale scenes and process RGB channels individually. Additionally, many motion blur datasets do not contain many samples that would allow robust training against over- and under-exposed scenes. To our knowledge, no dataset with sufficient motion blur captured at a high frame rate exists. Therefore, due to data limitations, we do not test against these lighting scenarios.

We could alternatively use an arbitrarily short exposure to mitigate motion deblurring and focus solely on denoising. However, shorter exposures suffer from quantization artifacts on the sensor and require extensive denoising algorithms [59], [60], [61], [62]. Thus, our method is a more robust solution to different lighting scenarios.

Future Work In future work, we would like to address the aforementioned limitations. Such work would include incorporating HDR scenes so that we can train for over- and under-exposed scenes. We also do not explicitly focus on optical blur or defocus blurs, and thus an improved E2E model could include modeling the optics for improved robustness against different types of blurs. It would be additionally useful to expand E2E methods like ours to programmable sensors that can capture different color channels as well. Different colors captured at different exposures such as that of Jiang et al. [14] can provide helpful cues in reconstruction.

Conclusion These efforts add to the growing foundation for emerging programmable sensors in computational imaging. As these sensors become more widespread in their use, we may begin to reframe our thinking of how to address ill-posed computer vision tasks, from object classification to HDR imaging. This work serves as a step in that direction.

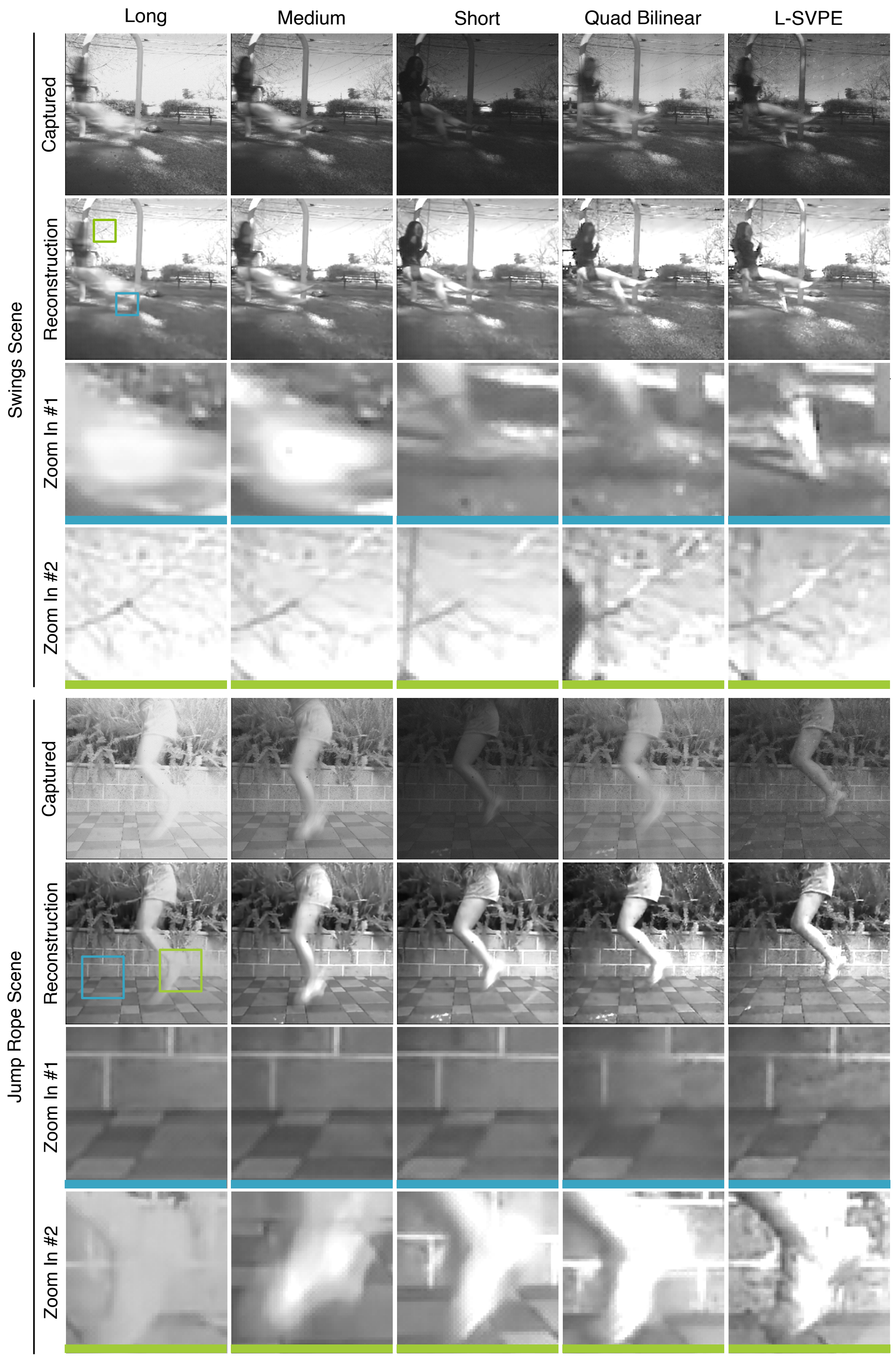


Fig. 7. Reconstructions of images captured using a physical focal-plane sensor–processor prototype (SCAMP-5). We compare reconstructions of two scenes (*Swings* and *Jump Rope*) from global exposures (*Long*, *Medium*, and *Short*) models and the best performing spatially varying exposures (*Quad* with Bilinear interpolation and L-SVPE). **Rows 1** and **5**: Captured images. The Short capture is noisier than its longer exposure counterparts. **Rows 2** and **6**: Reconstruction from the networks. **Rows 3** and **4**: Zoom ins of the reconstructed *Swings* scene. *L-SVPE* can successfully recover the lower shoe in addition to maintaining the high frequency detail of the static tree. **Rows 7** and **8**: Zoom ins of the reconstructed *Jump Rope*. *L-SVPE* preserves the sharp edges of the ground tiles and reconstructs the details of the shoe.

Acknowledgments

This project was in part supported by NSF Award 1839974, an NSF Graduate Research Fellowship (DGE-1656518), a PECASE by the ARL, and SK Hynix. We thank Piotr Dudek for providing the SCAMP-5 sensor and related discussions. The authors would also like to thank Mark Nishimura, Alexander Bergman, and Nitish Padmanaban for thoughtful discussions.

References

[1] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, M.-H. Yang, and L. Shao, "Multi-stage progressive image restoration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 821–14 831.

[2] S. Nah, T. Hyun Kim, and K. Mu Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3883–3891.

[3] M. Suin, K. Purohit, and A. Rajagopalan, "Spatially-attentive patch-hierarchical network for adaptive motion deblurring," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3606–3615.

[4] X. Tao, H. Gao, X. Shen, J. Wang, and J. Jia, "Scale-recurrent network for deep image deblurring," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8174–8182.

[5] D. Park, D. U. Kang, J. Kim, and S. Y. Chun, "Multi-temporal recurrent neural networks for progressive non-uniform single image deblurring with incremental temporal training," in *European Conference on Computer Vision*. Springer, 2020, pp. 327–343.

[6] M. Jin, G. Meishvili, and P. Favaro, "Learning to extract a video sequence from a single motion-blurred image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6334–6342.

[7] A. Levin, P. Sand, T. S. Cho, F. Durand, and W. T. Freeman, "Motion-invariant photography," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3, pp. 1–9, 2008.

[8] S. Elmalem, R. Giryes, and E. Marom, "Motion deblurring using spatiotemporal phase aperture coding," *Optica*, vol. 7, no. 10, pp. 1332–1340, 2020.

[9] E. Yosef, S. Elmalem, and R. Giryes, "Video reconstruction from a single motion blurred image using learned dynamic phase coding," *arXiv preprint arXiv:2112.14768*, 2021.

[10] R. Raskar, A. Agrawal, and J. Tumblin, "Coded exposure photography: motion deblurring using fluttered shutter," in *ACM SIGGRAPH 2006 Papers*, 2006, pp. 795–804.

[11] J. Gu, Y. Hitomi, T. Mitsunaga, and S. Nayar, "Coded rolling shutter photography: Flexible space-time sampling," in *2010 IEEE International Conference on Computational Photography (ICCP)*. IEEE, 2010, pp. 1–8.

[12] Y.-W. Tai, N. Kong, S. Lin, and S. Y. Shin, "Coded exposure imaging for projective motion deblurring," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 2408–2415.

[13] A. Agrawal and Y. Xu, "Coded exposure deblurring: Optimized codes for psf estimation and invertibility," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 2066–2073.

[14] Y. Jiang, I. Choi, J. Jiang, and J. Gu, "Hdr video reconstruction with tri-exposure quad-bayer sensors," *arXiv preprint arXiv:2103.10982*, 2021.

[15] S. J. Carey, A. Lopich, D. R. Barr, B. Wang, and P. Dudek, "A 100,000 fps vision sensor with embedded 535gops/w 256× 256 simd processor array," in *2013 Symposium on VLSI Circuits*. IEEE, 2013, pp. C182–C183.

[16] Á. Zarándy, *Focal-plane sensor-processor chips*. Springer Science & Business Media, 2011.

[17] M. Z. Wong, B. Guillard, R. Murai, S. Saeedi, and P. H. Kelly, "Analognet: Convolutional neural network inference on analog focal plane sensor processors," *arXiv preprint arXiv:2006.01765*, 2020.

[18] P. E. Debevec and J. Malik, "Recovering high dynamic range radiance maps from photographs," in *ACM SIGGRAPH 2008 classes*, 2008, pp. 1–10.

[19] J. N. Martel, L. K. Mueller, S. J. Carey, P. Dudek, and G. Wetzstein, "Neural sensors: Learning pixel exposures for hdr imaging and video compressive sensing with programmable sensors," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 7, pp. 1642–1653, 2020.

[20] H. M. So, J. N. Martel, P. Dudek, and G. Wetzstein, "Mantissacam: Learning snapshot high-dynamic-range imaging with perceptually-based in-pixel irradiance encoding," *arXiv preprint arXiv:2112.05221*, 2021.

[21] J. Sun, W. Cao, Z. Xu, and J. Ponce, "Learning a convolutional neural network for non-uniform motion blur removal," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 769–777.

[22] A. Chakrabarti, "A neural approach to blind motion deblurring," in *European conference on computer vision*. Springer, 2016, pp. 221–235.

[23] C. J. Schuler, M. Hirsch, S. Harmeling, and B. Schölkopf, "Learning to deblur," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 7, pp. 1439–1451, 2015.

[24] K. Purohit and A. Rajagopalan, "Region-adaptive dense network for efficient motion deblurring," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 11 882–11 889.

[25] H. Miao, W. Zhang, and J. Bai, "Aggregated dilated convolutions for efficient motion deblurring," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2018, pp. 1–6.

[26] Z. Tu, H. Talebi, H. Zhang, F. Yang, P. Milanfar, A. Bovik, and Y. Li, "Maxim: Multi-axis mlp for image processing," *arXiv preprint arXiv:2201.02973*, 2022.

[27] G. Wetzstein, A. Ozcan, S. Gigan, S. Fan, D. Englund, M. Soljacic, C. Denz, D. A. B. Miller, and D. Psaltis, "Inference in artificial intelligence with deep optics and photonics," *Nature*, vol. 588, pp. 39–47, 2020.

[28] V. Sitzmann, S. Diamond, Y. Peng, X. Dun, S. Boyd, W. Heidrich, F. Heide, and G. Wetzstein, "End-to-end optimization of optics and image processing for achromatic extended depth of field and super-resolution imaging," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–13, 2018.

[29] J. Chang and G. Wetzstein, "Deep optics for monocular depth estimation and 3d object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 10 193–10 202.

[30] H. Ikoma, C. M. Nguyen, C. A. Metzler, Y. Peng, and G. Wetzstein, "Depth from defocus with learned optics for imaging and occlusion-aware depth estimation," in *2021 IEEE International Conference on Computational Photography (ICCP)*. IEEE, 2021, pp. 1–12.

[31] Y. Wu, V. Boominathan, H. Chen, A. Sankaranarayanan, and A. Veeraraghavan, "Phasecam3d—learning phase masks for passive single view depth estimation," in *2019 IEEE International Conference on Computational Photography (ICCP)*. IEEE, 2019, pp. 1–12.

[32] C. A. Metzler, H. Ikoma, Y. Peng, and G. Wetzstein, "Deep optics for single-shot high-dynamic-range imaging," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1375–1385.

[33] Q. Sun, E. Tseng, Q. Fu, W. Heidrich, and F. Heide, "Learning rank-1 diffractive optics for single-shot high dynamic range imaging," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1386–1396.

[34] E. Nehme, D. Freedman, R. Gordon, B. Ferdman, L. E. Weiss, O. Alalouf, T. Naor, R. Orange, T. Michaeli, and Y. Shechtman, "Deepstorm3d: dense 3d localization microscopy and psf design by deep learning," *Nature methods*, vol. 17, no. 7, pp. 734–740, 2020.

[35] A. Sinha, J. Lee, S. Li, and G. Barbastathis, "Lensless computational imaging through deep learning," *Optica*, vol. 4, no. 9, pp. 1117–1125, 2017.

[36] S. S. Khan, V. Sundar, V. Boominathan, A. Veeraraghavan, and K. Mitra, "Flatnet: Towards photorealistic scene reconstruction from lensless measurements," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[37] M. Kellman, E. Bostan, M. Chen, and L. Waller, "Data-driven design for fourier ptychographic microscopy," in *2019 IEEE International Conference on Computational Photography (ICCP)*. IEEE, 2019, pp. 1–8.

[38] A. Chakrabarti, "Learning sensor multiplexing design through back-propagation," in *Advances in Neural Information Processing Systems*, 2016, pp. 3081–3089.

[39] E. Vargas, J. N. Martel, G. Wetzstein, and H. Arguello, "Time-multiplexed coded aperture imaging: Learned coded aperture and pixel exposures for compressive imaging systems," *arXiv preprint arXiv:2104.02820*, 2021.

[40] S. Su and W. Heidrich, "Rolling shutter motion deblurring," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1529–1537.

[41] H.-G. Jeon, J.-Y. Lee, Y. Han, S. J. Kim, and I. S. Kweon, "Multi-image deblurring using complementary sets of fluttering patterns," *IEEE Transactions on Image Processing*, vol. 26, no. 5, pp. 2311–2326, 2017.

[42] V. Rengarajan, S. Zhao, R. Zhen, J. Glotzbach, H. Sheikh, and A. C. Sankaranarayanan, "Photosequencing of motion blur using short and long exposures," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 510–511.

[43] M. Iliadis, L. Spinoulas, and A. K. Katsaggelos, "Deepbinarymask: Learning a binary mask for video compressive sensing," *arXiv preprint arXiv:1607.03343*, 2016.

[44] J. N. Martel, L. K. Müller, S. J. Carey, and P. Dudek, "High-speed depth from focus on a programmable vision chip using a focus tunable lens," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–4.

[45] J. Chen, S. J. Carey, and P. Dudek, "Feature extraction using a portable vision system," in *IEEE/RSJ Int. Conf. Intell. Robots Syst., Workshop Vis.-based Agile Auton. Navigation UAVs*, 2017.

[46] L. Bose, J. Chen, S. J. Carey, P. Dudek, and W. Mayol-Cuevas, "Visual odometry for pixel processor arrays," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4604–4612.

[47] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 881–892, 2002.

[48] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[49] S.-J. Cho, S.-W. Ji, J.-P. Hong, S.-W. Jung, and S.-J. Ko, "Rethinking coarse-to-fine approach in single image deblurring," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 4641–4650.

[50] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 136–144.

[51] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.

[52] Y. Blau and T. Michaeli, "The perception-distortion tradeoff," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6228–6237.

[53] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[54] H. Kiani Galoogahi, A. Fagg, C. Huang, D. Ramanan, and S. Lucey, "Need for speed: A benchmark for higher frame rate object tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1125–1134.

[55] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[56] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[57] L.-Y. Wei, "Multi-class blue noise sampling," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, pp. 1–8, 2010.

[58] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE transactions on image processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[59] C. Chen, Q. Chen, J. Xu, and V. Koltun, "Learning to see in the dark," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3291–3300.

[60] K. Wei, Y. Fu, J. Yang, and H. Huang, "A physics-based noise formation model for extreme low-light raw denoising," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2758–2767.

[61] M. Tassano, J. Delon, and T. Veit, "Fastdvdnet: Towards real-time deep video denoising without flow estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1354–1363.

[62] H. Jiang and Y. Zheng, "Learning to see moving objects in the dark," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7324–7333.

[63] B. Mildenhall, J. T. Barron, J. Chen, D. Sharlet, R. Ng, and R. Carroll, "Burst denoising with kernel prediction networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2502–2510.

[64] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian, "Practical poissonian-gaussian noise modeling and fitting for single-image raw-data," *IEEE Transactions on Image Processing*, vol. 17, no. 10, pp. 1737–1754, 2008.

[65] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[66] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learningt," *Cited on*, vol. 14, no. 8, p. 2, 2012.

[67] L. Zhang, A. Deshpande, and X. Chen, "Denoising vs. deblurring: Hdr imaging techniques using moving cameras," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 522–529.

Supplementary Material

A Network Details

A.1 Simulating noise

Given that we are simulating capture on a sensor, we take care to add shot noise, a Poisson process that is signal-dependent, and read noise, a Gaussian-approximated process independent of the signal. Following Mildenhall et al. [63], we model both noise sources as a Gaussian distribution:

$$\mathbf{y}_p \sim \mathcal{N}(\mathbf{x}_p, \sigma_r^2 + \sigma_s \mathbf{x}_p), \tag{6}$$

where $\mathbf{y}_p$ represents the noisy measurement of the true intensity $\mathbf{x}$ in pixel location p. We vary the noise parameters uniformly across images to simulate changes in sensor gain, and use the gain values as presented in [63] (σ_r = [1e-3, 3e-2], σ_s = [1e-4, 1e-2]). An accurate mixed noise model [64] allows us to more closely simulate sensor capture over models that use overly simplified additive Gaussian noise. We then clip all measurements to be within $[0, 1]$ to simulate sensor quantization.

We believe the interpolation step is, in fact, a bottleneck to achieving the model's full reconstruction potential. Just as one uses color-specific gradients in classic debayering of RGB images, one can imagine that a similar exposure-specific interpolation can be beneficial here. We note an exposure-specific interpolation algorithm may work better than our naive Bilinear and Scatter interpolation. However, the design of such an algorithm is beyond the scope of this paper.

A.2 Decoding U-Net

The 2D U-Net used takes as input C-channel images of resolution 512×512 to produce a single channel output of the same resolution as its input. The architecture used consists of 6 downsampling blocks and 6 upsampling modules. Each downsampling block consists of a single `Conv` operation with 3×3 kernels, followed by a `ReLU`. The convolution uses a striding of size 1 and padding of size 1. The upsampling module uses `ConvTranspose` with kernel size 4×4, stride size 2, and padding size 1. This is followed by 2 `Conv` operations with the same initialization as the those of the downsampling operation. We then apply a final `Conv` operation with kernel size 1×1. We found that `BatchNorm` decreased overall performance.

A.3 Using alternative decoders

We sought decoder alternatives that were comparably less demanding of memory and computation than the U-Net. We attempted training a model with the first two stages of MPRNet [1], as the entire network did not fit on a single 24 GB GPU. We found that the MPRNet performed comparably well as the U-Net, but required five times as much memory and thus was not our network of choice.

As part of our ablation study, we use DnCNN [58]. The head consists of a `Conv` operation with a 3×3 kernel, with stride size 1 and padding size 1, followed by a `ReLU`. The body consists of 15 "blocks," each consisting of a `Conv` with the same parameters as the head, followed by a `BatchNorm` with momentum 0.9 and a `ReLU`. The tail consists of a single `Conv` operation. The network is structured to predict the residual of the input, and we subtract the predicted residual from the 4th channel. This channel corresponds to the shortest channel in the Quad exposure.

B Training Details

B.1 Learning a discretized exposure

Our model discretizes the exposure time per pixel so that it can be programmed in-pixel to our focal-plane sensor–processor, the SCAMP-5 [15]. Each pixel is limited to 7 analog memories and 13 single-bit digital memories. We use 6 of the digital memories to create $2^6 = 64$ possible time-slots, where 64 out of 64 "on" encoded bits would be equivalent to our Long exposure (32 for Medium and 8 for Short). We segment the 64 time-slots into 8 to get 8 learnable options for exposure times. Here, 8 out of 8 is Long, 4 out of 8 is Medium, and 1 out of 8 would be a Short exposure length. Learning an exposure length with options 1 through 8 poses a challenge for neural networks, as we are interested in regressing an integer value which is non-differentiable. For this, we use a straight-through estimator (STE) [65], [66] that allows us to convert our learned exposure values to integer values in the forward function and pass the gradients to these exposure values in the backward pass. Using a STE also allows us to initialize our learned exposure value, contrary to using methods like Gumbel-Softmax which injects Gumbel noise at every iteration.

B.2 Training

We trained using batch sizes of 2 due to memory constraints on GPUs available for our experiments. We also use `ReduceLROnPlateau` as a learning rate scheduler with a patience of 20 and factor of 0.8 for a gradual decrease in learning. To speed up training, we initialize L-SVPE with the Quad exposure arrangement.

B.3 Dataset

We segment the videos in the NfS dataset [54] into segments each containing eight frames for two reasons: (1) the average of eight 240-fps frames makes for a single 30 fps which is common in many consumer cameras and (2) the SCAMP-5 [15], is limited to 13 digital memories (e.g. $2^1 3$ time points where the shutter can be "on" or "off"). We choose to use 6 digital memories to get 64 time slots which can be divided into eight frames easily.

We found that the first frame as ground truth improved reconstruction more than using the middle or last frame. We argue this is due to the fact that denoising has been observed to be simpler than blind deblurring [67]. When we have a mixture of short and longer exposures, the network can focus on denoising a short exposure to match the ground truth and using static pixels from the longer exposures to supplement the appearance of the short exposure.

We also tested our method on the GoPro Motion Blur dataset [2]. However, we found that given that the dataset was created by deliberately creating motion via rotation of the camera axis, many of the training videos only had global motion. In cases of extensive global motion, the long exposure becomes less helpful in reconstruction. We were interested in have a more generalized method towards global, local, and no motion and thus found the NFS dataset more optimal for our purposes.

C Experiment Details

C.1 Baselines

We now discuss in detail our reasoning behind the design of our baselines. Note that **B** represents Bilinear interpolation and **S** represents Scatter interpolation.

C.1.1 Global Exposures

- **Burst Average**: We perform a per-pixel averaging of all the frames in the scene, equivalently to a Long exposure or a 30 fps capture, which performs temporal denoising but not spatial denoising. This baseline is subject to misalginment if there is a dynamic motion. The averaged image is then used to compute metrics, skipping network reconstruction completely. This baseline serves as the result of not using any deep learning.
- **Short**: We use the first frame of the video segment input as the short exposure (240 fps). In this case, the decoding network simply must scale and denoise the image, as the ground truth is also the first frame. We found that denoising in general can be easier than deblurring. However, low light instances will prompt the network to blur to denoise, losing critical high frequency detail.
- **Medium**: We average the first four frames to simulate a medium exposure (120 fps), an intermediate choice of possible lengths.
- **Long**: Much like the Burst Average, we perform a per-pixel average of all 8 frames (30 fps), but we then use a decoder for refinement and spatial denoising.
- **Full**: This baseline is a stack of the Short, Medium, and Long exposure at full resolution all captured with the same start time and same viewpoint, which is physically impossible to capture but easy to simulate.

Given an 8-frame long scene input, we sought to test a number of global exposures that could be representative of not only ones you may capture on a camera but also could be feasibly implemented on the SCAMP-5 sensor. We believe testing each individual Short, Medium, and Long exposure lengths would give us a sufficient grasp of how the network performs on different exposure lengths, and found that, as expected, they provided a linear relationship in performance. The longer the exposure, the more susceptible the exposure was to misalignment, and the worse the reconstruction became. The Full exposure is physically infeasible to capture at full resolution, but we believe that the combination of the Short, Medium, and Long exposures at full resolution would serve as useful theoretical upper bound. This baseline demonstrates the utility of information from additional exposures. Based on Table C1, we see that the Full is able to outperform any individual global exposure by themselves.

C.1.2 Coded Exposures

- **Uniformly Random (S)**: We uniformly sample each pixel between exposure lengths 1 to 8. We chose a uniformly random array to demonstrate the worst-case scenario of using multiple exposures. The spatial variability of the exposure would make it challenging for the decoding network to learn a spatially invariant kernel to apply across the measurement. We use PyTorch's `torch.randint` function to randomly select integers from 1 to 8 to populate a empty 512×512 array. We do not use length 0 because a pixel that is "off" does not provide any additional information for reconstruction.
- **Poisson Random (S)**: This baseline serves as a non-ideal scenario similar to the Uniform Random exposure due to the spatial variability. However, the Poisson sampling guarantees that in every 3×3 grid of the 512×512 array, there is an equal likelihood of finding a pixel of exposure lengths 1 through 8. We believe that having a more equal distribution of the different lengths would be useful for interpolating exposures that were more reflective of the entire scene. We use a multi-class Poisson disk sampling algorithm [57] to populate an empty 512×512. Given the dense sampling of our array, there will be remaining unfilled pixels that do not meet the radial constraint of the multi-class Poisson sampling. These remaining pixels (we usually had $\sim$20 unfilled pixels out of a total of 512×512 pixels) are manually filled in to maintain an even distribution of exposure lengths.
- **Nonad (nine-tuple) (B, S)**: This baseline serves as a regular arrangement of the full spectrum of exposure lengths. The learned decoding kernel of the decoding network could then be applied spatially invariantly. We use PyTorch's `torch.randperm` to randomly arrange an array of pixel exposures, each pixel a unique length from 1 through 8, with an additional exposure length 1 to make a total of 9 pixels in the 3×3. We then tile this fixed arrangement. We add a short exposure pixel as the last pixel due to the ease of denoising the short exposure over deblurring.
- **Quad (B, S)**: This baseline is designed to resemble the work closest to ours from Jiang et al. [14]. Notably, they use a custom decoder network for their reconstruction. However, we were interested in using a decoder network that could be applied fairly well to other baselines with less parameters so we opt for the U-Net. Following Jiang et al. [14], we use a fixed tile coded arrangement of LMMS (Long-Medium-Medium-Short). We then tile this fixed arrangement across the 512×512 sensor.

We compare the Scatter and Bilinear interpolation of these coded exposures where applicable to demonstrate the utility of the interpolation step in terms of perceptual quality. We provide the full numerical results from the main paper in Table C1, which uses our default parameters including our custom perceptual loss. We also provide qualitative comparisons of all the baselines in Figure C1. To generate RGB images, we run each channel through the

TABLE C1
Quantitative results of all baselines. *Full*, our theoretical upper bound performs the best on all three metrics, and *L-SVPE* performs the best out of all other baselines on all metrics.

Model	Interpolation	PSNR↑	SSIM↑	LPIPS↓
Burst Average	—	25.305	0.827	0.462
Short	—	33.618	0.949	0.158
Medium	—	28.669	0.919	0.165
Long	—	25.666	0.879	0.215
Uniform Random	Scatter	31.607	0.946	0.139
Poisson Random	Scatter	32.098	0.950	0.136
Nonad	Scatter	31.614	0.943	0.142
Nonad	Bilinear	32.731	0.954	0.125
Quad [14]	Scatter	33.272	0.957	0.122
Quad	Bilinear	33.632	0.962	<u>0.111</u>
L-SVPE	Scatter	<u>34.383</u>	<u>0.963</u>	<u>0.111</u>
Full	—	**36.375**	**0.976**	**0.085**

decoding network trained on grayscale images. We find that L-SVPE generalizes across different color channels best, after Full, in addition to reconstructing high frequency detail and deblurring.

D Prototype

The physical prototype, a SCAMP-5 focal-plane sensor–processor [15], we used has a non-linear camera response function (CRF). We first calibrate the CRF using a set of exposures and the CRF function from OpenCV. We use the CRF to linearize our captures before processing.

The SCAMP-5 has a sensor resolution of 256×256. However, we trained all of our models at 512×512 resolution. Thus, to reconstruct the images captured from the SCAMP-5, we pad these images with zeros before in putting them into our network. We then apply an sRGB curve ($\gamma = 2.0$) to the reconstruction for display.

The supplemental videos of the scene show the frame-by-frame reconstruction of the *Swings* and *Jump Rope* scene. Although the capture itself is reflective of the fps designated for each model, the recording is saved at 60 fps due to the design of the camera software. We also slow down the video to 5 fps to allow for viewing detail.

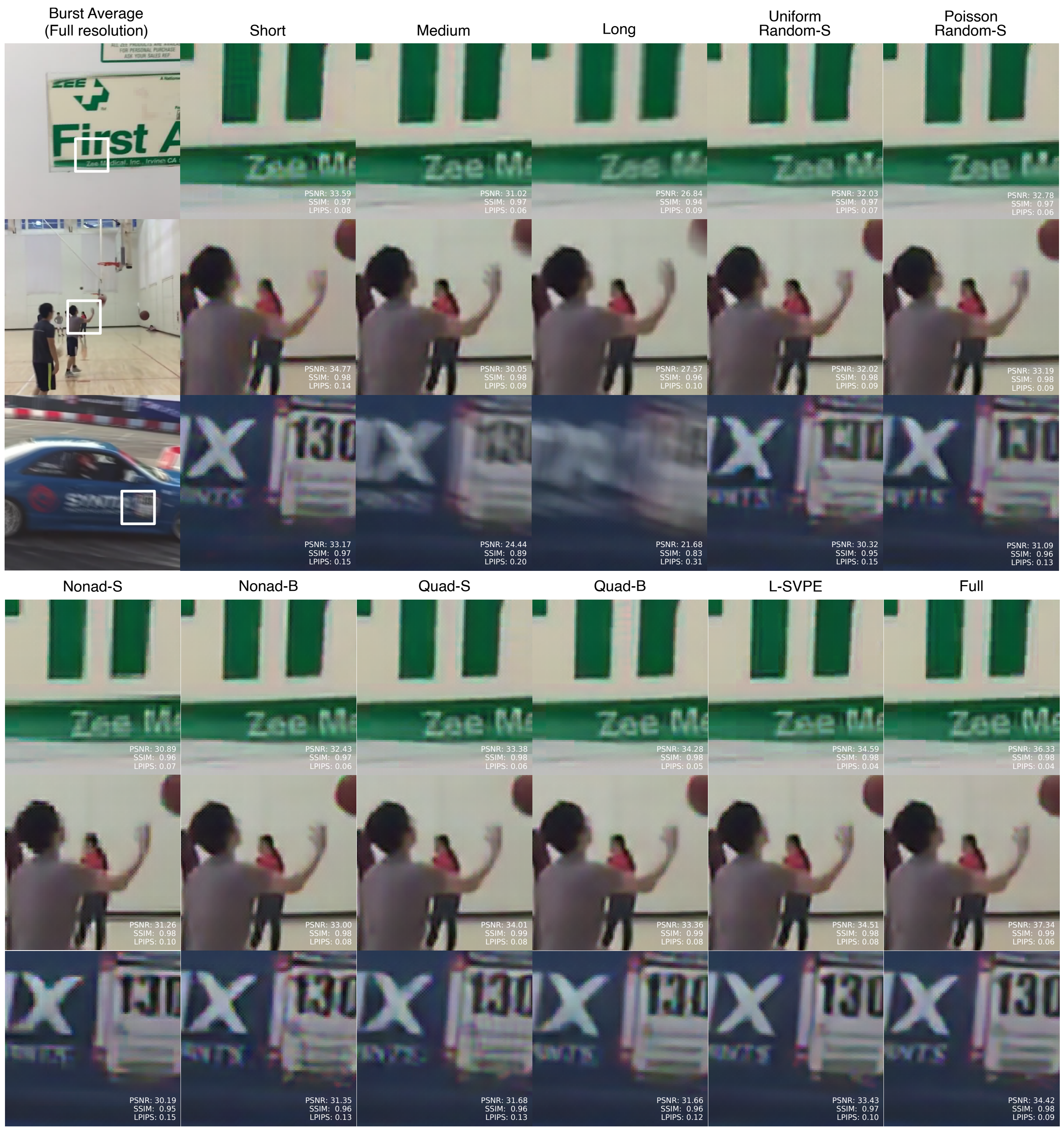

Burst Average (Full resolution)
Short
Medium
Long
Uniform Random-S
Poisson Random-S
PSNR: 33.59 SSIM: 0.97 LPIPS: 0.08
PSNR: 31.02 SSIM: 0.97 LPIPS: 0.06
PSNR: 26.84 SSIM: 0.94 LPIPS: 0.09
PSNR: 32.03 SSIM: 0.97 LPIPS: 0.07
PSNR: 32.78 SSIM: 0.97 LPIPS: 0.06
PSNR: 34.77 SSIM: 0.98 LPIPS: 0.14
PSNR: 30.05 SSIM: 0.98 LPIPS: 0.09
PSNR: 27.57 SSIM: 0.96 LPIPS: 0.10
PSNR: 32.02 SSIM: 0.98 LPIPS: 0.09
PSNR: 33.19 SSIM: 0.98 LPIPS: 0.09
PSNR: 33.17 SSIM: 0.97 LPIPS: 0.15
PSNR: 24.44 SSIM: 0.89 LPIPS: 0.20
PSNR: 21.68 SSIM: 0.83 LPIPS: 0.31
PSNR: 30.32 SSIM: 0.95 LPIPS: 0.15
PSNR: 31.09 SSIM: 0.96 LPIPS: 0.13
Nonad-S
Nonad-B
Quad-S
Quad-B
L-SVPE
Full
PSNR: 30.89 SSIM: 0.96 LPIPS: 0.07
PSNR: 32.43 SSIM: 0.97 LPIPS: 0.06
PSNR: 33.38 SSIM: 0.98 LPIPS: 0.06
PSNR: 34.28 SSIM: 0.98 LPIPS: 0.05
PSNR: 34.59 SSIM: 0.98 LPIPS: 0.04
PSNR: 36.33 SSIM: 0.98 LPIPS: 0.04
PSNR: 31.26 SSIM: 0.98 LPIPS: 0.10
PSNR: 33.00 SSIM: 0.98 LPIPS: 0.08
PSNR: 34.01 SSIM: 0.99 LPIPS: 0.08
PSNR: 33.36 SSIM: 0.99 LPIPS: 0.08
PSNR: 34.51 SSIM: 0.98 LPIPS: 0.08
PSNR: 37.34 SSIM: 0.99 LPIPS: 0.06
PSNR: 30.19 SSIM: 0.95 LPIPS: 0.15
PSNR: 31.35 SSIM: 0.96 LPIPS: 0.13
PSNR: 31.68 SSIM: 0.96 LPIPS: 0.13
PSNR: 31.66 SSIM: 0.96 LPIPS: 0.12
PSNR: 33.43 SSIM: 0.97 LPIPS: 0.10
PSNR: 34.42 SSIM: 0.98 LPIPS: 0.09

Fig. C1. Qualitative comparison of baselines. Each row of individual images (carried over to create two larger rows) shows an example frame of the average of all the video segment frames (*Burst Average*) at full resolution and a 100×100 crop of a region for all baselines. **Row 1:** An example of a little to no motion scene. The lettering is most clearly recovered and free of color artifacts with *L-SVPE* and *Full*. **Row 2:** An example of a scene with local motion. The reconstruction of the man's hand and the deblurring of the basketball is best observed in spatially varying coded exposures. *L-SVPE* is also relatively free of color artifacts. **Row 3:** An example of a scene with global motion. The "X" on the side of the car and the lines on the logo next it are clearly recovered by *L-SVPE* while methods like *Short* and *Quad* introduce more blurring artifacts.