

Washington University in St. Louis

## Washington University Open Scholarship

---

Engineering and Applied Science Theses &  
Dissertations

McKelvey School of Engineering

---

Summer 8-15-2021

### Reasoning about Scene and Image Structure for Computer Vision

Zhihao Xia

*Washington University in St. Louis*

Follow this and additional works at: [https://openscholarship.wustl.edu/eng\\_etds](https://openscholarship.wustl.edu/eng_etds)



Part of the [Computer Sciences Commons](#)

---

#### Recommended Citation

Xia, Zhihao, "Reasoning about Scene and Image Structure for Computer Vision" (2021). *Engineering and Applied Science Theses & Dissertations*. 667.

[https://openscholarship.wustl.edu/eng\\_etds/667](https://openscholarship.wustl.edu/eng_etds/667)

This Dissertation is brought to you for free and open access by the McKelvey School of Engineering at Washington University Open Scholarship. It has been accepted for inclusion in Engineering and Applied Science Theses & Dissertations by an authorized administrator of Washington University Open Scholarship. For more information, please contact [digital@wumail.wustl.edu](mailto:digital@wumail.wustl.edu).

WASHINGTON UNIVERSITY IN ST. LOUIS

School of Engineering & Applied Science  
Department of Computer Science and Engineering

Dissertation Examination Committee:

Ayan Chakrabarti, Chair

Tao Ju

Brendan Juba

Ulugbek Kamilov

Kalyan Sunkavalli

Reasoning about Scene and Image Structure for Computer Vision

by

Zhihao Xia

A dissertation presented to  
The Graduate School  
of Washington University in  
partial fulfillment of the  
requirements for the degree  
of Doctor of Philosophy

August 2021  
St. Louis, Missouri



© 2021, Zhihao Xia

# Table of Contents

<b>List of Figures</b> .....	v
<b>List of Tables</b> .....	xii
<b>Acknowledgments</b> .....	xv
<b>Abstract</b> .....	xix
<b>Chapter 1: Introduction</b> .....	1
1.1 Scene map estimation from images .....	3
1.1.1 Quality of camera measurements .....	3
1.1.2 Ill-posed nature of scene map estimation .....	4
1.1.3 High-dimensional output and search space.....	6
1.1.4 Insufficient data for deep learning .....	6
1.2 Dissertation overview .....	7
<b>Chapter 2: Exploiting internal structures in low-light photography</b> .....	10
2.1 Related work.....	13
2.2 Exploiting patch similarity for image denoising .....	14
2.2.1 Introduction.....	14
2.2.2 Proposed denoising algorithm .....	16
2.2.3 Experiments.....	22
2.2.4 Discussion.....	29
2.3 Exploiting self-similarity in denoising kernels for burst photography .....	30
2.3.1 Introduction.....	30
2.3.2 Method .....	32
2.3.3 Kernel-based burst denoising .....	33
2.3.4 Experiments.....	37

2.3.5	Discussion.....	46
2.4	Exploiting scene appearance in flash photography .....	47
2.4.1	Introduction .....	47
2.4.2	Proposed approach .....	50
2.4.3	Experimental setup.....	55
2.4.4	Evaluation .....	57
2.4.5	Discussion.....	62
<b>Chapter 3: Learning without direct supervision for computational photog-</b>		
<b>raphy .....</b>		<b>64</b>
3.1	Training image estimators without ground-truth images .....	66
3.1.1	Introduction .....	66
3.1.2	Related work .....	69
3.1.3	Proposed approach .....	71
3.1.4	Experiments .....	76
3.1.5	Discussion.....	80
3.2	Training a dark flash normal camera without ground-truth normals.....	84
3.2.1	Introduction .....	84
3.2.2	Related work.....	86
3.2.3	Network design and training.....	89
3.2.4	Dataset .....	93
3.2.5	Evaluation .....	96
3.2.6	Discussion.....	100
<b>Chapter 4: Probabilistic scene map estimation for modular inference .....</b>		<b>102</b>
4.1	Related work.....	105
4.2	Probabilistic monocular depth .....	108
4.2.1	Proposed method .....	108
4.2.2	Monocular inference tasks .....	111
4.2.3	Experimental results .....	112
4.3	Depth estimation with additional information .....	115
4.3.1	Proposed approach .....	115

4.3.2	Applications.....	117
4.3.3	Experimental results .....	119
4.3.4	Analysis and ablation.....	123
4.4	Discussion .....	125
<b>Chapter 5: Conclusion .....</b>		<b>126</b>
<b>References .....</b>		<b>129</b>

# List of Figures

Figure 1.1:	Visualization of the system considered in this dissertation. An image (b) represents a two-dimensional projection of incident light from a scene (a), as captured by a camera—often with noise and other degradation. We study the inverse problem—inferring scene properties from the captured (b) image with neural networks, where scene properties is represented by a dense array of real-valued numbers, i.e., (c) “scene maps”. (d) Example scene maps include high-quality images, depth maps, albedo, etc. This problem is ill-posed. Because the observed (b) image could correspond to multiple (e) plausible scenes. Scene examples are based on Adelson and Pentland’s “workshop” metaphor [1].	2
Figure 1.2:	Scene maps (for example, this depth map from the NYUv2 dataset [151]) are not arbitrary arrays of independent numbers. They exhibit statistical and spatial structure.....	5
Figure 2.1:	(a) A photo of New York City at night taken with a mobile phone (cr. Ayan Chakrabarti). (b) Human portraits taken with the night sight mode on Google Pixel 3 (cr. Alexander Schiffhauer [144]). (c) Astrophotography with a Huawei P30 Pro (cr. Justin Ng [125]).....	11
Figure 2.2:	Overview of Our Approach to Patch Denoising. We produce estimates of clean patches by weighted averaging across a candidate set of nearby patches in the observed noisy input. We decompose every patch using a wavelet and de-correlating color transform into sets of sub-band coefficients, with coefficients at the same scale, orientation, and color channel grouped together in each set. We then train a neural network that, given a pair of patches, computes a vector of matching scores—one for each group of coefficients. For every patch, we compute these score vectors with respect to all candidate patches. The denoised patch is obtained by averaging, across all candidates, of each group of coefficients using its corresponding matching scores.....	16

Figure 2.3:	Matching Network Architecture. To produce the matching scores $m_{ij}^g$ , we first extract a feature vector for all patches by passing the image through a feature extraction network, comprised of a set of convolutional layers with skip connections (where the join is performed by a concatenate operation. Then, for any pair of patches, we take the corresponding pair of feature vectors, and pass them after concatenation through a series of fully-connected layers. The final layer has a sigmoid activation, yielding scores that lie between 0 and 1. ....	19
Figure 2.4:	Example Crops of Denoised Images ( $\sigma = 50$ ). Compared to state-of-the-art denoising algorithms (IRCNN [185], DnCNN [184], and FFDNet [186]), we see that our overall method is often able to recover texture and detail with higher fidelity, by exploiting similar patterns in the input image itself. ....	25
Figure 2.5:	Effect of training set size. We report average PSNR on Urban-100 [64] for denoising at $\sigma = 50$ with our method and IRCNN [185], when both are trained with a limited number of training images (“Full” represents using the entire training set for our method, and the official model for IRCNN). While our estimates are always more accurate than those from IRCNN, the gap is especially higher when the number of training images is small. ....	26
Figure 2.6:	Visualization of Matching Score Distributions in Different Sub-bands. We show reference patches (indicated by blue squares) along with their local search windows from various images of the training set (9 windows per image), and visualize the matching scores predicted by our network. We show the predicted weights averaged across all sub-bands, as well as specific to different scales (averaging over color and orientation at each scale), and orientations (averaging over color and scale).....	28
Figure 2.7:	( <i>top</i> ) Qualitative result of our denoising network. The proposed approach (b) recovers fine geometric details from a very noisy image burst (a, only one frame is shown). ( <i>Bottom</i> ) our per-burst bases can compactly represent a large variety of kernels by exploiting the redundancy of local image structures. We clustered the per-pixel coefficients predicted from the noisy burst using $k$ -means. The clusters we obtain show strong spatial structure (d). For instance, we can identify a cluster corresponding to horizontal image edges (e, orange), one corresponding to vertical edges (e, green), and another for homogeneous regions with no structure (e, gray).....	30

Figure 2.8:	Our basis prediction network takes as input a burst of noisy input frames (a) together with the noise parameters (b). The frames are encoded into a shared feature space (c). These features are then decoded by two decoders with skip connections into a burst-specific basis of 3D kernels (e) and a set of per-pixel mixing coefficients (d). Both the coefficients and basis kernels are individually unit-normalized. Finally, we obtain per-pixel kernels by mixing the basis elements according to the coefficients and we apply them to the input burst to produce the final denoised image (f). .....	33
Figure 2.9:	We illustrate denoising performance on a benchmark synthetic grayscale test set [120] for our method, a direct prediction network (which directly regresses denoised pixels), and two KPN variants [115, 120] with the same kernel size $K = 15$ as our method. The numbers in inset refer to the PSNR (dB) on the full image. In addition to better quantitative performance, our method does better at reproducing perceptual details like textures, edges, and text. ....	38
Figure 2.10:	We visualize a few 3D kernels predicted by our approach (with $K = 15$ ), and those produced by standard KPN (with $K = 5$ and $K = 15$ ). For kernels predicted at a given location, we also show crops of the different noise-free frames centered at that point, with the support of the kernel marked in blue. In comparison to those from KPN, our kernels are more evenly distributed across all frames in the burst, with spatial patterns that closely follow the apparent motion in the burst. ....	45
Figure 2.11:	We show examples of color denoising using our method on our synthetic color test set, comparing these to direct prediction and our color-extended version of KPN [120] (with $K = 5$ ). Numbers refer to PSNR (dB) on the full image. ....	46
Figure 2.12:	Given a pair of images of low-light scenes captured with and without a flash (left), our method produces a high-quality image of the scene under ambient lighting (right). This output is generated by filtering the no-flash image with a predicted field of kernels—to capture a smoothed stimate of scene appearance under ambient lighting, followed by multiplication with a scale map that introduces high-frequency detail illuminated by the flash. ....	48
Figure 2.13:	<b>System Overview.</b> The denoising network takes as input a pair of flash, no-flash images (a) together with the noise parameters (b). After encoding, the resulting features (c) are decoded into a multi-scale basis (d), a set of pixel-wise coefficients (e) and a scale map (f). The no-flash image is filtered using the reconstructed kernels (g) and multiplied by the scale map to produce the final denoised output (h). ....	51

Figure 2.14:	<b>Performance vs. misalignment.</b> We show the performance profile of our method and select baselines as a function of average displacement between the two frames. Our model consistently delivers superior performance and is robust to large misalignment between its inputs.	59
Figure 2.15:	<b>Qualitative comparison.</b> Our method uses flash/no-flash image pairs to denoise low-light images. It produces cleaner outputs than baseline flash/no-flash denoisers ( <i>Direct (F+NF)</i> , <i>BPN (F+NF)</i> ), as well as single-image ( <i>Only No-Flash Input</i> ) and burst denoisers ( $2\times$ <i>No-Flash Burst</i> ). We also visualize our intermediate filtered no-flash image and scale map. ....	60
Figure 2.16:	<b>Qualitative comparison (continued).</b> Our method uses flash/no-flash image pairs to denoise low-light images. It produces cleaner outputs than baseline flash/no-flash denoisers ( <i>Direct (F+NF)</i> , <i>BPN (F+NF)</i> ), as well as single-image ( <i>Only No-Flash Input</i> ) and burst denoisers ( $2\times$ <i>No-Flash Burst</i> ). We also visualize our intermediate filtered no-flash image and scale map. ....	61
Figure 2.17:	<b>Flash vs. no-flash as reference frame.</b> We use the ambient-only image as the reference frame for our reconstruction ( <i>top</i> ), i.e. the ground truth is aligned to the no-flash image. We found this choice leads to a lower error on average, compared to the alternative, using the flash as reference ( <i>bottom</i> ). ....	61
Figure 2.18:	<b>Benefit of large kernels.</b> By using a 2-scale kernel decomposition, where the low-pass component is bilinearly upsampled, our model ( <i>top</i> ) can better denoise the ambient-only image. This leads to reduced residual chroma noise, which makes the scale map more effective at recovering fine details. Without it ( <i>bottom</i> ), the kernels are too small to effectively denoise the ambient image, so the scale map needs to compensate for the residual mid-frequency noise.....	62



Figure 3.1:	<b>Unsupervised Training from Measurements.</b> Our method allows training image estimation networks $f(\cdot)$ from sets of pairs of varied measurements, but without the underlying ground-truth images. ( <i>Top Right</i> ) We supervise training by requiring that network predictions from one measurement be consistent with the other, when measured with the corresponding parameter. ( <i>Bottom</i> ) In the blind training setting, when both the image and measurement parameters are unavailable, we also train a parameter estimator $g(\cdot)$ . Here, we generate a proxy training set from the predictions of the model (as it is training), and use synthetic measurements from these proxies to supervise training of the parameter estimator $g(\cdot)$ , and augment training of the image estimator $f(\cdot)$ . . . . .	67
Figure 3.2:	Images reconstructed by various methods from compressive measurements (at 10% ratio). . . . .	77
Figure 3.3:	Blind face deblurring results using various methods. Results from our unsupervised approach, with both non-blind and blind training, nearly match the quality of the supervised baseline. . . . .	82
Figure 3.4:	Image and kernel predictions on validation images. We show outputs of our model’s kernel estimator, that is learned as part of blind training to compute swap- and self-measurement losses. . . . .	83
Figure 3.5:	Estimating surface geometry from a single RGB image is challenging. We augment this input with a single NIR “dark flash” image captured at the same time, and present a network that can estimate high quality normal maps and reflectance maps (not shown) under a wide range of visible lighting conditions. . . . .	85
Figure 3.6:	Our network learns to estimate shape and reflectance from a single front-lit NIR image, a single RGB image under arbitrary lighting, and a semantic segmentation map computed from the RGB image (inputs are enclosed by the red line). During training we also use a stereo depth map and replace the RGB image under arbitrary lighting with 4 RGB+NIR image pairs captured under calibrated point lights (the training inputs are inside the blue dashed line). . . . .	87
Figure 3.7:	Illustration of our network and training strategy. We estimate network weights that minimize a photometric loss, computed between images rendered from our network outputs and ground truth images captured under known lighting, and a stereo loss, driven by differences between the output normals and those estimated using an independent stereo technique. . . . .	89

Figure 3.8:	Our hardware setup consists of controllable NIR and visible spectrum light sources, an RGB camera, a stereo pair of NIR cameras, and two NIR dot projectors. One of the NIR cameras and the RGB camera are aligned with a beamsplitter and all of these components are triggered electronically to record the types of images shown in Figure 3.6.....	94
Figure 3.9:	Impact of the photometric loss term in our training procedure and the Blinn-Phong BRDF in our image formation model, respectively. When trained without photometric loss, our network learns to output the stereo normals, which lack fine-scale details. This has a fairly small effect on the error measures in Table 3.3, but is perceptually significant as seen in these “n dot l” shading renderings. Our full image formation model, which includes a Blinn-Phong specular term, produces more accurate albedos across the face than using a Lambertian model alone.	94
Figure 3.10:	Comparison of our network to a modified version that takes only a single RGB image (“RGB Only”) as input. Example results for three common challenging lighting conditions. Top to bottom: low light / noisy inputs; mixed light colors; harsh directional lighting with saturated intensities. The “RGB only” network struggles to produce stable normal and reflectance estimates from these inputs in contrast to our method. ....	95
Figure 3.11:	Stereo methods often struggle to recover fine-scale surface details. <i>Left:</i> Applying a guided bilateral filter to raw stereo depths yields a smoother surface but with distorted features (e.g. the nose is reduced and skin wrinkles are missing). <i>Right:</i> We use the method of Nehab et al. [123] to compute a refined surface according to normals estimated with our method. Note how details are better preserved around the eyes, nose, and mouth, along with fine wrinkles and creases. ....	99
Figure 3.12:	Our method can be used to simulate adding lights to a scene to fill in shadows. ....	100
Figure 4.1:	Overview of our approach. Given an input color image, we use a common task-agnostic network to output a joint probability distribution $p(\mathbf{Z} \mathbf{I})$ over the depth map—formed as a sample approximation using outputs of a conditional VAE that generates plausible estimates for depth in overlapping patches. The mean of this distribution represents a standard monocular depth estimate, but the distribution itself can be used to solve a variety of inference tasks in different application settings—including leveraging additional depth cues to yield improved estimates. All these applications are enabled by a common model, <i>that is trained only once</i> . ....	104

Figure 4.2:	Generating samples with a conditional VAE. Our network generates samples for depth independently in each overlapping patch, and we run it multiple times to generate multiple plausible samples per-patch. The input to the VAE comes from pre-trained feature extraction layers from a state-of-the-art monocular model [49]. Samples generated for different patches (including those that overlap) are kept statistically independent—after conditioning on the image—by using separate per-patch latent vectors.....	109
Figure 4.3:	Example depth estimates for different applications. We show outputs from our method for both the pure monocular setting, as well as the improved estimates we obtain combining our distributional output with additional depth information—such as different kinds of partial measurements, and user guidance with annotation and selection. ....	120
Figure 4.4:	Analysis of distributional output and inference method on the test set. Our distribution allows for many possible global depth explanations, visualized here by choosing one of the generated samples in each patch based on the rank of its accuracy going from best (oracle) to worst (adversary), and computing global depth by overlap-average. These solutions span a large range in accuracy, and without any additional information, the mean monocular estimate lies in the middle of this range. But when additional cues are available, they can be effectively exploited by our <i>MAP</i> estimation method to extract better solutions from our distribution. ....	124

# List of Tables

Table 2.1:	Denoising Performance at Various Noise Levels on Different Datasets. We report performance in terms of Average PSNR (dB) and SSIM. To gauge robustness, we also report the 25 <sup>th</sup> %-ile worst-case PSNR (dB), computed across $8 \times 8$ patches across each dataset. Ours-Blind refers to results from a common model of our method that is trained for a range of noise levels $\sigma \in [0, 55]$ (and does not have knowledge of the specific noise level of its input). . . . .	22
Table 2.2:	Window Size and Pre-Training Ablation. We report average PSNR (db) of the initial match-averaged estimates from our method on a validation set for $\sigma = 25$ . Run-times are for $256 \times 256$ images on a 1080Ti GPU. . . . .	27
Table 2.3:	Denoising performance on a synthetic grayscale benchmark [120]. We report performance in terms of Average PSNR (dB). Following [115, 120], our BPN was not trained on the noise levels implied by the gain in the fourth column. Numbers for KPN* ( $K = 5$ ) and MKPN* are based on our implementation of these techniques. Numbers for all other methods, including the end-to-end regression model to directly synthesize denoised pixel intensities (denoted as Direct), are from [120]. Our method widely outperforms all prior methods at all noise levels. . . . .	37
Table 2.4:	Ablation study on our validation dataset. Performance is reported in terms of Average PSNR (dB). Beyond motivating our parameter choices ( $K = 15, B = 90$ ), this demonstrates that our use of a burst-specific spatio-temporal basis outperforms standard KPN [120], separable spatial kernels, a common spatial basis for all burst frames, separate spatial bases per-frame, and a fixed, input-agnostic basis. All these variants were trained with the same settings ( $K = 15, B = 90$ ) as our model. . . . .	40

Table 2.5:	Average basis rank for each noise level (first row), average rank of the union of two bases from random burst pairs (second row), and the average overlap ratio (third row) between the subspaces spanned by the two bases. The low overlap justifies our prediction of a burst-specific basis. ....	43
Table 2.6:	FLOPS and runtimes on $1024 \times 768$ resolution images for different KPN denoising approaches. All variants of our basis prediction network are significantly faster than KPN and match the compute cost of separable filters (with better denoising quality). Increasing the kernel size for our technique comes at marginal cost thanks for the Fourier filtering approach. This allows us to use large kernels for better denoising performance.....	43
Table 2.7:	Denoising performance on our synthetic color test set. We report performance in terms of Average PSNR (dB). Numbers for KPN* ( $K = 5$ ) are based on our implementation of [120]. Our method outperforms KPN by more than 1 dB at all noise levels. ....	44
Table 2.8:	<b>Quantitative results.</b> Thanks to the richer signal provided by the flash input, our method outperforms our single image denoising baseline, and a 2-frame burst denoising baseline. Comparisons to standard burst denoising approaches adapted to use flash–no-flash pairs show that our model architecture with its filtering/scale decomposition and larger kernels outperforms previous work. These results hold over a wide range of ambient light levels, shown here as dimming factors between the low-light no-flash input and a well-lit ground-truth target. ....	57
Table 2.9:	<b>Ablation study.</b> We compare the performance of our method to two ablations. One uses the flash image instead of the no-flash image as reference for the geometric transformation. The other uses a kernel basis without interpolation, leading to an effective kernel size of only $15 \times 15$ . ....	59
Table 3.1:	Performance (in PSNR dB) of various methods for compressive measurement reconstruction, on BSD68 and Set11 images for different compression ratios.....	77
Table 3.2:	Performance of various methods on blind face deblurring on test images from [146]. ....	81
Table 3.3:	Mean absolute angular error in degrees of normal maps computed with modified versions of our full network. Results are reported for the five lighting conditions described in Section 3.2.5.....	97

Table 4.1:	Results our probabilistic output for monocular depth estimation on the NYUv2 test set. Methods that we compare to are specifically proposed for monocular depth estimation.....	113
Table 4.2:	RMS error for depth estimation from different numbers of sparse measurements, when making measurements at random locations vs. with guidance from our distribution. Given the measurements, we use our depth estimation algorithm described in Section 4.3.2 in both cases	113
Table 4.3:	Error rates for pairwise ordinal depth ordering from our common model, compared to other methods that used accurate ordering as an objective during training. We also report baseline errors from predictions just based on our mean depth estimate. ....	114
Table 4.4:	Part A of results for various applications on the NYUv2 test set. We use distributional outputs from our common model to generate depth estimates in a diverse variety of application settings when different forms of additional depth cues are available. We compare to other methods for these applications, including those (shaded background) dependent on task-specific networks trained separately for each setting. Our network, in contrast, is task-agnostic and trained only once. ....	121
Table 4.5:	Part B of results for various applications on the NYUv2 test set. We use distributional outputs from our common model to generate depth estimates in a diverse variety of application settings when different forms of additional depth cues are available. We compare to other methods for these applications, including those (shaded background) dependent on task-specific networks trained separately for each setting. Our network, in contrast, is task-agnostic and trained only once. ....	122
Table 4.6:	Ablation study on validation set. We evaluate different ways of generating samples: using a GAN instead of a VAE, and using different patch-sizes $p$ (with proportional strides $s$ ). For each case, we compare achievable accuracy of individual samples via the “oracle” estimate (see Figure 4.4), vs. their utility for actual inference—in the pure monocular case and with random sparse measurements ( $\#100$ ). We also evaluate the importance of patch overlap by considering larger strides for our chosen model.....	125

# Acknowledgments

My PhD journey started with the first computer vision class I went to, which till this day I still remember vividly. A picture of that class was recorded and locked in my brain. In the center of that picture is a slide of a robot holding a watering can, while trying to figure out the color of the flower and how far the plant is. Standing next to that slide, it was Ayan talking passionately about what computer vision is. I was immediately struck by the exciting mission of computer vision and Ayan's vibrant intellectual energy. Ever since that moment, my PhD research has been all about working with Ayan to enable that robot, or any computer, to perceive the world as we human beings. In retrospect, I feel immeasurably glad that I made that decision. Ayan has taught me how to do research, write papers, give talks and most importantly how to think. For the past four years, he has been my go-to person whenever I have a question and he can give me advice on almost everything. I will certainly miss him as an inspiring advisor who led me into the door of computer vision, but I am also deeply looking forward to more conversations that I will have with him as a friend in the future.

One of the privileges that I enjoyed most as a researcher, especially a computer vision researcher, is my collaborations with industrial labs, through which I have the chance to work with more amazing mentors: Federico Perazzi, Michaël Gharbi, Kalyan Sunkavalli, Supreeth Achar and Jason Lawrence. I could not ask for a better host for my first internship than

Federico: he is a wonderful mentor and a fun friend. Michaël has given me so many advice and has certainly become the role model for my next few years. I always enjoy talking to Kalyan for his intellectual insights and deep knowledge. I'm grateful to Supreeth for showing me how to set up cameras and lights to collect data. Jason is always passionate and so fun to talk to, and he showed me to think bold and that our research can really change our life.

Besides research, my friends and peers are the reason why my PhD journey has been so fun and they are: Wei Tang, Rusi Yan, Xiaojian Xu, Kyle Singer, Jeffery Jung, Aaron Park. I spent so much time talking to Wei, about research and almost everything in life. Countless times we talked for so long that Rusi has to remind us it is midnight. I'm so glad that we have grown together from two naive students to two independent researchers. I spent more New Year's Eves and enjoyed more delicious meals with Kyle and Xiaojian than with my parents in the last four years. I had many fun and exciting trips with Jeff and Aaron and they've become my wonderful memories. I'm also grateful to Chen Liu for being a great friend and giving me so much help and advice. I'm also fortunate to have many chances to talk about papers and research with my peers. They are Huayi Zeng, Dan Zeng, Xingyi Du and Yu Sun.

These few words go to my family. My Mom and Dad have being given me unconditional supports for my entire life. Even though they do not understand my research, I find every bit of my work related to them because they have made the person that I am today. My little brother, Yihao (I named him when he was born although he now has a English name Justin), I hope that one day when you can understand this and when you are reading this, you have found the thing that you love as I have right now. I am so happy that I can call Bei my family, for that we have been together for 8 years through our entire undergrad and PhD. I simply, cannot imagine my life without her. There are so many amazing journeys ahead



waiting for us. Oh and of course thanks to our little cat, Panghu or Chubby tiger. Ever since we adopted her, she has been on the slides of almost every talk that I gave.

Finally, I want to dedicate this thesis to my Grandma (Nainai). The darkest moment of my PhD was when I was sitting on the hotel floor in Aspen for WACV and crying alone after I heard that she was taken to the ICU. She passed away in a few months. My Grandma raised me when I was a child, and the last time I saw her when she was conscious, she told me to stay well. This is for you, Nainai.

Zhihao Xia

*Washington University in Saint Louis*

*August 2021*

To grandma

## ABSTRACT OF THE DISSERTATION

Reasoning about Scene and Image Structure for Computer Vision

by

Zhihao Xia

Doctor of Philosophy in Computer Science

Washington University in St. Louis, 2021

Professor Ayan Chakrabarti, Chair

The wide availability of cheap consumer cameras has democratized photography for novices and experts alike, with more than a trillion photographs taken each year. While many of these cameras—especially those on mobile phones—have inexpensive optics and make imperfect measurements, the use of modern computational techniques can allow the recovery of high-quality photographs as well as of scene attributes.

In this dissertation, we explore algorithms to infer a wide variety of physical and visual properties of the world, including color, geometry, reflectance etc., from images taken by casual photographers in unconstrained settings. We specifically focus on neural network-based methods, while incorporating domain knowledge about scene structure and the physics of image formation. We describe novel techniques to produce high-quality images in poor lighting environments, train scene map estimators in the absence of ground-truth data and learn to output our understanding and uncertainty on the scene given observed images.

The key to inferring scene properties from casual photography is to exploit the internal structure of natural scenes and the expressive capacity of neural networks. We demonstrate that neural networks can be used to identify the internal structure of scenes maps, and that

our prior understanding on natural scenes can shape the design, training and the output representation of neural networks.

# Chapter 1

## Introduction

The long history of human using photography to record where they are and what they see at an instant, *i.e.*, a *scene*, dated back to the 4th century BCE when the Chinese philosopher Mozi described the earliest version of a pinhole camera. Projecting the world onto a 2D plane, whether it being a wall, a Kodak photographic film or an array of CMOS sensors, what a camera captures is far beyond a grid of pixels ever since then. When a human looks at a photograph, our brain sees what the scene was like at the moment of capture. For example, when we see an image full of people, our human perception system can often easily tell the actual color of a person's hair, which person is standing closer, the material of a person's clothes, etc. Empowering a computer to do the same, namely, decode scene properties from a single or a few images, however, is very challenging and has long been a research problem in computer vision and computational photography.

In the last two decades, drastic changes have taken place in the way we record a scene and decode a photograph. Cameras have become cheap and are now available on mobile phones. As a result, many more photographs are being taken by both professional photographers and

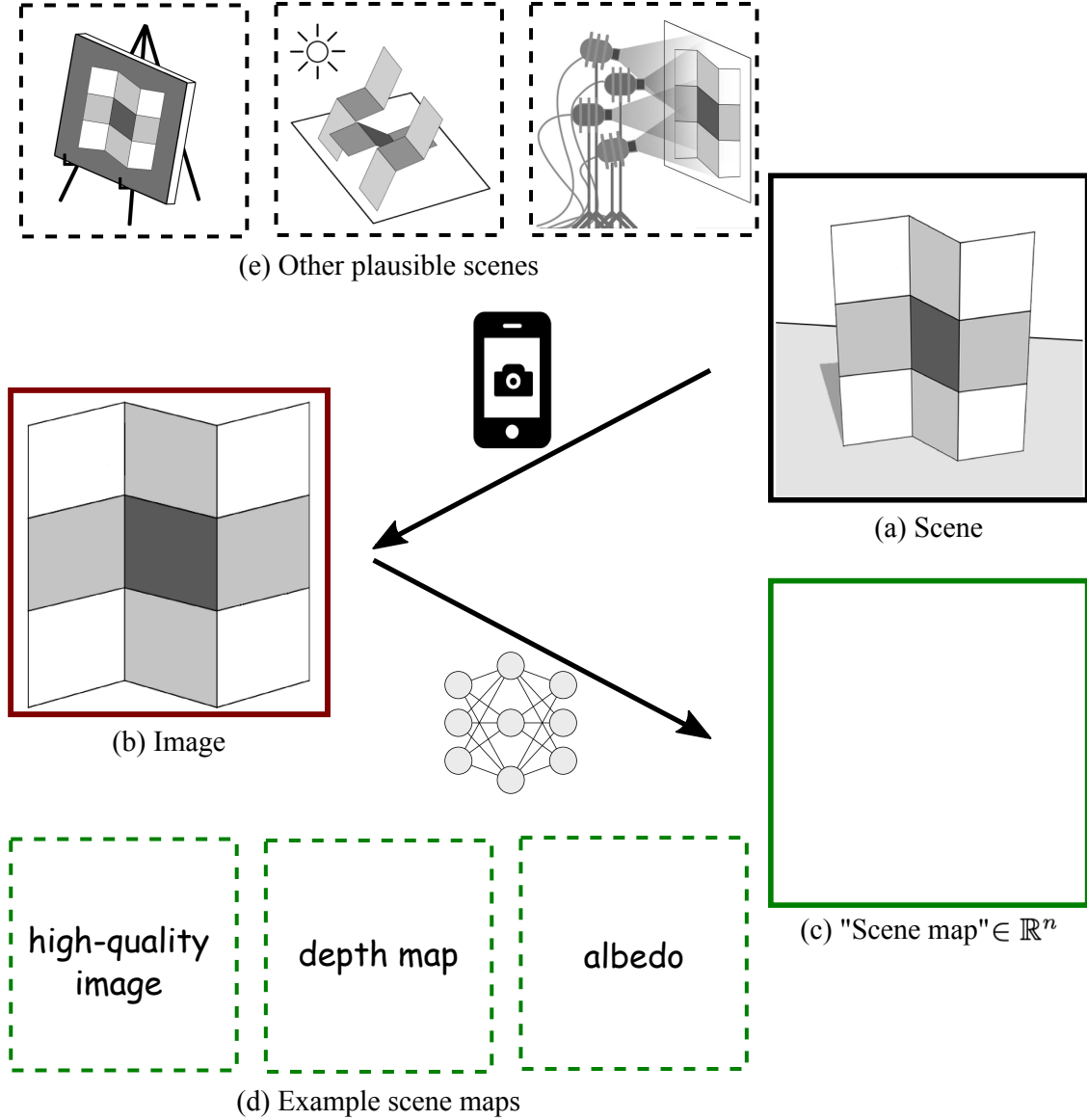


Figure 1.1: Visualization of the system considered in this dissertation. An image (b) represents a two-dimensional projection of incident light from a scene (a), as captured by a camera—often with noise and other degradation. We study the inverse problem—inferring scene properties from the captured (b) image with neural networks, where scene properties is represented by a dense array of real-valued numbers, i.e., (c) “scene maps”. (d) Example scene maps include high-quality images, depth maps, albedo, etc. This problem is ill-posed. Because the observed (b) image could correspond to multiple (e) plausible scenes. Scene examples are based on Adelson and Pentland’s “workshop” metaphor [1].

novice users. As a result, there are many more applications—in photography, image editing, scene understanding, and beyond—where computer vision algorithms can find immediate

use. However, these algorithms must deal with the fact that many of the measurements that these cameras make—especially with small and inexpensive optics—are imperfect, and that there is much more diversity in scene content, illumination conditions, camera viewpoint, etc. in casual photography.

Thankfully, the availability and ease of collecting large amounts of data has allowed the use of large scale machine learning models. In particular, deep neural networks (DNNs) have revolutionized computer vision. After seeing initial successes in semantic tasks such as image classification and object detection, they are being increasingly used for tasks in computational photography and for the recovery of physical scene properties. However, these tasks are far from being solved. Many challenges, as we discuss next, still remain.

## 1.1 Scene map estimation from images

In this dissertation, we study the problem of scene map estimation from casually captured images (visualized in 1.1). Specifically, we define a *scene map* to be a high-dimensional dense array of continuous-valued scalar or vectors at each pixel on the image plane, which characterize different physical and visual properties of the scene. Example scene maps include image intensities for image restoration, depth maps and surface normals for geometry reasoning, reflectance and shading in photometric applications, optical flow in video analysis. The goal of scene map estimation tasks is to solve the inverse problem to infer scene properties from observed images.

### 1.1.1 Quality of camera measurements

Casual photography, as opposed to professional photography and cinematography, involves images taken by regular people with consumer grade cameras. While casual photography

provides photographers the advantage of convenience to capture the scene spontaneously, the unsatisfying lighting environment, the poorly-tuned camera settings and the limitation of mobile cameras often lead to unsatisfying image quality. The undesirable effects of captured photos, including observation noise and artificial blur, obscures the encoded information about the scene and creates great challenges for scene map estimation. Many of my dissertation researches are indeed about using computational models to produce a unique kind of scene map, *i.e.* high quality images, from degraded measurements. The other alternative is to develop robust models to directly estimate other scene properties from images with unsatisfying quality (as done in Section 3.2).

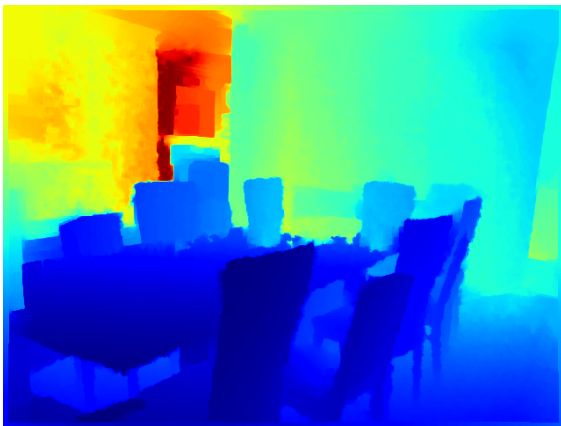
### 1.1.2 Ill-posed nature of scene map estimation

The inverse problem of scene map estimation is ill-posed due to the confounding of a variety of variables in the image formation model. Given an observed image, there are often multiple or even infinite physically plausible solutions to explain the captured image (see Figure 1.1). The ill-posed nature of scene map estimation is the fundamental challenge in many tasks. Image restoration seek to recover the underlying clean image from observational artifacts. Intrinsic decomposition methods must disambiguate shading variation or reflectance variation that lead to an appearance change. Depth estimation methods often struggle to infer the absolute depth of the scene because of the ambiguity of scale and distance.

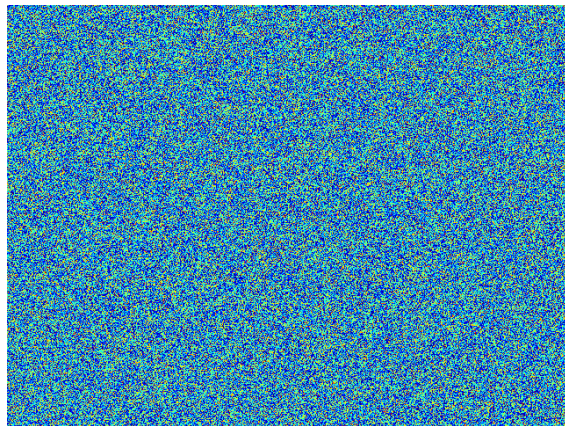
Fortunately, scene maps are not arbitrary arrays of independent numbers and exhibit statistical and spatial structure. An easy way to see this is that the human visual system can easily distinguish a real scene map or image from arbitrary arrays, or even shuffled versions of true maps, even when one is not familiar with the scene the image corresponds to. An example is shown in Figure 1.2. Therefore, researchers have tried to exploit the statistics and the structure of natural scene maps to identify the most natural estimation from all plausible



solutions. Such prior knowledge are often defined as statistical models on natural scene maps and treated as regularization or cost function on the probability of a candidate estimation. Examples include different types of sparsity constraint on natural images and piece-wise constant variations for albedo maps [6].



(a) A depth map



(b) Randomly shuffled version of (a)

Figure 1.2: Scene maps (for example, this depth map from the NYUv2 dataset [151]) are not arbitrary arrays of independent numbers. They exhibit statistical and spatial structure.

The last decade has witnessed significant strides in computer vision through the use of convolutional neural networks (CNNs). While initially used for semantic applications with discrete outputs, extensive works have shown that CNNs can also be useful for physics-based and image-level scene map estimation. By training on a large-scale dataset of natural scene maps, CNN based methods take a different approach and learn to automatically choose the inverse mapping (from images to scene maps) from a space of possible solutions (also often referred as *hypothesis* and *hypothesis space*). However, how to come up with a *hypothesis space* for scene map estimations, which is defined by the architectural design of a neural network, still remains an active research problem. A main contribution of this dissertation is to combine the power of learning from large-scale datasets with our prior knowledge on natural scene maps to tackle the ill-posed nature of scene map estimation.

### 1.1.3 High-dimensional output and search space

A unique challenge of scene map estimation is the high-dimensionality of both the dense array of real-valued estimation and the search space of these estimations. High-dimensionality not only makes inference more challenging, but also increases the computational burden of neural networks, which often prevents neural-based solution being deployed in practice. To alleviate the curse of high-dimensionality, we must again recall that natural scene maps are not arbitrary, the manifold of natural scene maps only spans a small subspace of the entire space of high-dimensional arrays. As we will show in Chapter 2, by regularizing the hypothesis space of neural networks with our knowledge on the internal structure of natural scene maps, we can achieve more effective and more efficient neural network based inference models.

In many tasks, we need to model the probability distribution of a scene map estimation which is often intractable due to its high dimensionality. We cannot, on the other hand, ignore the correlation between estimates for different pixels as we must model the joint statistics and structure of the scene map. Therefore, we often take a patch-wise approach and divide scene maps into overlapping patches (as shown in Section 2.2 and Chapter 4). The inter-dependencies of different patches are characterized by their overlapping regions.

### 1.1.4 Insufficient data for deep learning

Unlike semantic tasks, *e.g.*, image classification, where a large-scale dataset with ground-truth categorical labels can be easily collected through crowd sourcing, collecting ground-truth scene maps is extremely hard and sometimes even impossible for most computational photography applications. For example, acquiring a dataset of high-resolution normal maps with enough diversity and at scale is still a problem as of today. Even very well defined tasks such as

image restoration require laborious work and expert knowledge and camera setup to capture ground-truth images. Insufficient data, unfortunately, has been a major obstacle for the implementation of neural network based models in real-world applications. After all, it is often those tasks where acquiring scene maps is expensive that we need scene estimators the most.

The other end of the spectrum lies the synthetic dataset, which has been widely adopted in the academia due to the lack of high-quality real dataset. However, while synthetic datasets can be useful to test new ideas and to augment real datasets, the unavoidable domain gap between real and synthetic images means that they are insufficient by themselves—i.e., real data is still required during training to achieve satisfactory performance during inference. In Chapter 3, we dive into this problem and study how to train neural networks for computational photography in the absence of direct supervision.

## 1.2 Dissertation overview

In this dissertation, we address the aforementioned challenges of scene map estimation. The dissertation is presented as follows:

**Chapter 2** describes three neural network based methods for different image capturing strategies to produce natural looking, noise-free images in low-light environments. We exploit the internal structure of various forms of inputs for low-light photography, including natural images, image bursts and flash/no-flash pairs. Novel neural network architectures are presented, based on our prior knowledge on the input signals, to achieve state-of-the-art performance in each case.

**Chapter 3** proposes to take advantages of our understanding on the observation model or the image formation of given inference tasks, and train scene map estimators in the absence of ground-truth scene maps. We demonstrate that, with carefully chosen regularization and training scheme, even indirect measurements (which are far easier to collect) of the scene alone could provide sufficient supervision to train a neural network to produce high-quality estimates of scene properties.

**Chapter 4** takes a different direction by acknowledging that scene estimators will never be perfect and instead rely on the recourse of other sources of information about the scene that are often available in practice (*i.e.*, other measurements of the scene ). We first train a neural network to output a rich probabilistic representation to encode our understanding and uncertainty of the scene given the captured image. The network is versatile as its output can be combined with a variety of types of additional inputs to yield improved scene map estimations without retraining.

We conclude the dissertation in **Chapter 5** and discuss potential avenues for future work.

Note that early versions of this research have appeared in the following publications.

- **Zhihao Xia** and Ayan Chakrabarti. “Training Image Estimators without Image Ground-Truth”, *NeurIPS*, 2019 (**Spotlight**)
- **Zhihao Xia** and Ayan Chakrabarti. “Identifying Recurring Patterns with Deep Neural Networks for Natural Image Denoising”, *WACV*, 2020.
- **Zhihao Xia**, Federico Perazzi, Michaël Gharbi, Kalyan Sunkavalli, Ayan Chakrabarti. “Basis Prediction Networks for Effective Burst Denoising with Large Kernels”, *CVPR*, 2020.

- **Zhihao Xia**, Patrick Sullivan, Ayan Chakrabarti. “Generating and Exploiting Probabilistic Monocular Depth Estimates”, *CVPR*, 2020 (**Oral**).
- **Zhihao Xia**, Michaël Gharbi, Federico Perazzi, Kalyan Sunkavalli, Ayan Chakrabarti. “Deep Denoising of Flash and No-Flash Pairs for Photography in Low-Light Environments”, *CVPR*, 2021.
- **Zhihao Xia**, Jason Lawrence, Supreeth Achar. “A Dark Flash Normal Camera”, *arXiv*, 2020.

## Chapter 2

# Exploiting internal structures in low-light photography

Insufficient light is the fundamental challenge for photographers to produce a high-quality image. The noise inherent in the measured process dominates the captured image, leading to a low signal-to-noise ratio for low-light photography. Direct solutions to gather more light include expanding the camera's aperture, increasing the exposure time and adding more lights. However, many of these solutions are not feasible due to the physical limit of both the camera and the environment in practice, especially for mobile phone cameras which is the most popular type of cameras nowadays. The thickness and size constraints of mobile phones limit the size of aperture and the length of optical path. Long exposure time often leads to the undesired effect of blur especially for scenes with moving objects and hand-held photography. Despite these challenges, great advances have been made in both the industry and the research community, to produce impressive low-light photographs with mobile phone cameras, which were previously deemed impossible (See Figure 2.1).



Figure 2.1: (a) A photo of New York City at night taken with a mobile phone (cr. Ayan Chakrabarti). (b) Human portraits taken with the night sight mode on Google Pixel 3 (cr. Alexander Schiffhauer [144]). (c) Astrophotography with a Huawei P30 Pro (cr. Justin Ng [125]).

These results are achieved with computational algorithms. The search for computational models to produce high-quality images in low-light inspired many interesting research problems, from single-image denoising, one of the most fundamental problem in image processing, to the more recent problem of burst denoising. Unfortunately, image denoising is fundamentally ill-posed. A denoising model must be able to synthesize contents because of this ill-posed nature, therefore must rely on the structure of contents in natural images. However, the sheer diversity of content, that can be present in photographs of natural scenes, makes them a challenge for any denoising algorithms as they must model their statistics.

Many early approaches in single-image denoising are inspired by the discovery of internal redundancy in natural images. While the content of different images can be very diverse, patches within an natural image are often similar and recur frequently. However, the state-of-the-art single-image denoising algorithms take a different approach, by relying on the expressive capacity of convolutional neural networks and the “external statistics” from a large-scale dataset of images. In parallel to this is the so-called computational photography, which tackles low-light photography by proposing a novel capturing strategy— capturing more than one frame to produce a single high-quality image. These frames can be images with

the same appearance but small misalignments due to scene and camera motion, or ambient and flash pair before and after the flash of the camera is shoot. The internal structure of such captured signals has a new meaning, and is perhaps more prominent, as it not only characterizes the contents the scene, but also the relations and the invariance of different frames. Unlike single-image denoising, very few works have tried to leverage machine learning to merge these frames.

In this chapter, we study how to produce a noise-free image in low-light environments with different capturing strategy and different forms of inputs. We focus on exploiting the internal structure of these inputs, whether them be the content within a single image, or the structure of different frames. Unlike previous approaches, we combine machine learning with our knowledge on the internal structure of input signals. On one hand, we leverage the expressive capacity of neural networks to identify the structure and patterns that present in our inputs. On the other hand, we embed our prior knowledge on what natural images should be like into the design of our neural networks.

The rest of the chapter is presented as follows. We begin by reviewing prior works in low-light photography in Section 2.1. We then study the classical problem of single-image denoising by exploiting self-similarity in patches within a natural image in Section 2.2. In Section 2.3, we discuss how to leverage self-similarity to design an efficient and effective burst denoising neural network. Finally, we present our approach to produce a noise-free image from a flash/no-flash pair with the ambient color and mood in Section 2.4.



## 2.1 Related work

**Image Denoising.** Early works reduced image noise using regularization schemes like sparse-coding [90] and low-rank factorization [56] to model the local statistics of natural images. Other classical approaches have exploited the recurrence of natural image patterns, averaging pixels with similar local neighborhoods [15, 37, 100, 130, 139, 156, 178]. Current state-of-the-art denoisers use deep neural networks. Burger *et al.* [18] were the first to show the ability of even shallow multi-layer perceptrons to outperform traditional methods such as BM3D [32], and more recent approaches utilizing deeper networks and complex architectures [102, 172, 184, 185, 190] have since led to further improvements in reconstruction accuracy.

**Burst Image Denoising.** Burst imaging can achieve impressive denoising results, by capturing multiple frames in quick succession. Recent burst denoising algorithms have focused on circumventing the frame misalignment that exists in a real burst. Some methods estimate pixel-wise displacement [57, 58, 59, 74, 109]; others only require coarse global registration, and rely on neural networks to account for the remaining displacement. Amongst the latter group, kernel prediction networks (KPNs) proposed by Mildenhall *et al.* [120] predicts per-pixel kernels that are then applied to the input burst to produce the denoised output. They demonstrate that KPNs outperform direct pixel synthesis networks that produce oversmooth results. Subsequent work has extended this idea to use multiple kernels of varying sizes at every pixel [115].

**Flash Denoising.** Flash photography enables the capture of low-noise images in low-light environments using short exposure times and low ISO settings. But, the additional source flash light drastically changes the mood of the scene captured. To remedy this while benefiting

from the flash image’s higher signal-to-noise ratio, several approaches have used the flash as reference to denoise a noisy ambient (no flash) image. Petschnigg *et al.* [131] and Eisemann *et al.* [41] use the flash photo to guide a joint-bilateral filter that denoises the ambient image, transferring high-frequency content from the flash photo. Krishnan and Fergus [78] exploit the correlations between dark flash images and visible light to denoise the ambient image and restore fine details. Yan *et al.* [175] combine gradients from the flash image with the no-flash image for denoising. These methods all use hand-crafted heuristics to decide which image features to preserve from each of the flash and no-flash inputs.

More recent work [92, 159] have replaced these heuristics with deep neural networks. Li *et al.* [92] use the (aligned) flash photo as guidance to denoise ambient images. Wang *et al.* [159] address some of the shortcomings of dark flash photography by adding a stereo RGB image to the capture setup. After being registered and aligned the two images are fused using recent techniques for hyperspectral image restoration and fast image enhancement [25, 51].

## 2.2 Exploiting patch similarity for image denoising

### 2.2.1 Introduction

In this section, we study the simplest form of low-light photography: single-image denoising. Our goal is to recover an estimate of a clean image from a noisy observation with additive white Gaussian noise (AWGN).

An important class of single-image denoising methods adopt an internal modeling approach, to exploit self-similarity in images by relying on their “internal statistics” [16, 32, 33]. A particularly successful example in this class is the BM3D [32, 33] algorithm, which identifies sets of similar patches in noisy images using sum of squared distances (SSD) as the matching

metric, and then uses the statistics of each set to denoise patches in that set. Applying this process twice, BM3D produced high-quality estimates that, until recently, represented the state-of-the-art in image denoising performance.

However, recent methods have been able to exceed this performance by using neural networks trained to regress to clean image patches from noisy ones [18, 27, 185]. With carefully chosen architectures, these methods are able to use the powerful expressive capacity of neural networks to better learn and encode image statistics from external databases, and thus exceed the capability of self-similarity based methods. In this section, we describe a single-image denoising method that brings the expressive capacity of neural networks to the task of identifying and leveraging recurring patterns in the underlying images from their noisy observations.

We introduce a novel matching network that looks at pairs of noisy patches at a time, and makes fine-grained predictions of the similarity of their underlying clean versions. Specifically, our network outputs separate matching scores for different groups of wavelet coefficients, to exploit similarities that exist at some orientations and scales, but not others. These scores are used for averaging to form an initial denoised estimate. We propose a two-step process to train this matching network, with respect to denoising quality, that leads to convergence to a better network model. We feed the initial match-averaged estimates, along with the original noisy image, as input to a standard regression-based denoising network to produce the final denoised estimate.

Extensive experiments on multiple datasets show that our method consistently yields higher quality estimates than the state-of-the-art on a variety of metrics. Indeed, even a *blind* version of our model—that does not have knowledge of the noise level—outperforms state-of-the-art

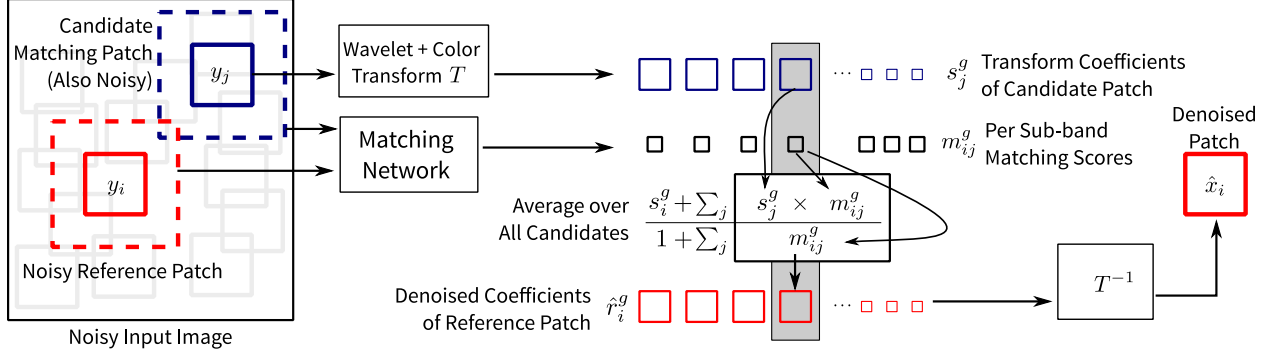


Figure 2.2: Overview of Our Approach to Patch Denoising. We produce estimates of clean patches by weighted averaging across a candidate set of nearby patches in the observed noisy input. We decompose every patch using a wavelet and de-correlating color transform into sets of sub-band coefficients, with coefficients at the same scale, orientation, and color channel grouped together in each set. We then train a neural network that, given a pair of patches, computes a vector of matching scores—one for each group of coefficients. For every patch, we compute these score vectors with respect to all candidate patches. The denoised patch is obtained by averaging, across all candidates, of each group of coefficients using its corresponding matching scores.

methods that do. Finally, we show that our method has a distinct advantage over regression-based networks when learning from only a small amount of training data. In these cases, our method is able to generalize better due to its reliance on per-image internal statistics.

### 2.2.2 Proposed denoising algorithm

Our goal is to produce an estimate of an image  $X$  given observation  $Y$  that is degraded by i.i.d. Gaussian noise, i.e.,

$$Y = X + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma_z^2 I). \quad (2.1)$$

Our algorithm leverages the notion that many patterns will occur repeatedly in different regions in the underlying clean image  $X$ , while the noise in those regions in  $Y$  will be un-correlated and can be attenuated by averaging. In this section, we describe our approach to training and using a deep neural network to identify these recurring patterns from the noisy image, and forming an initial estimate of  $X$  by averaging matched patterns. We then

use a second network to regress to the final denoised output from a combination of these initial estimates and the original noisy observation.

**Denoising by Averaging Recurring Patterns.** Our initial estimate is formed by denoising individual patches in the image, by computing a weighted average over neighboring noisy patches with weights provided by a matching network. Formally, given the noisy observation  $Y$  of an image  $X$ , we consider sets of overlapping patches  $\{y_i = P_i Y\}$  (corresponding to clean versions  $\{x_i = P_i X\}$ ), where each  $P_i$  is a linear operator that crops out intensities of a different square patch (of size  $8 \times 8$  in our implementation) from the image. We then use a de-correlating color space followed by a Harr wavelet transform to obtain coefficients  $s_i = T y_i$  (corresponding to clean versions  $r_i = T x_i$ ), where the orthonormal matrix  $T$  represents the color and wavelet transforms.

We group these coefficients into sets  $\{s_i^g\}$  where each set includes all coefficients with the same orientation (horizontal, vertical, or diagonal derivative), scale or pyramid level, and color channel<sup>1</sup>. Then, for every patch  $y_i$  we consider a set of candidate matches composed of other noisy patches in the image  $y_j, j \in \mathcal{N}_i$ , from a large neighborhood around  $i$ . As illustrated in Figure 2.2, our method produce an estimate of the denoised coefficients  $\hat{r}_i$  as a weighted average of the corresponding coefficients of the candidate patches:

$$\hat{r}_i^g = \left( 1 + \sum_{j \in \mathcal{N}_i} m_{ij}^g \right)^{-1} \left( s_i^g + \sum_{j \in \mathcal{N}_i} m_{ij}^g s_j^g \right), \quad (2.2)$$

where  $m_{ij}^g \geq 0$  are scalar matching weights that are a prediction of the similarity between the  $g^{th}$  set of coefficients in patches  $i$  and  $j$  respectively.

---

<sup>1</sup>For  $8 \times 8$  patches, this gives us 30 coefficient groups: 27 corresponding to 3 color channels, 3 scales, and 3 derivative orientations; and an additional 3 coefficients for the scaling coefficients of the 3 color channels.

This gives us a denoised estimate for each patch  $\hat{x}_i$  in the image as  $T^{-1}\hat{r}_i$ . We then obtain an estimate  $\hat{X}$  of the full clean image simply by averaging the denoised patches  $\hat{x}_i$ , i.e., the denoised estimate of each pixel is computed as the average of its estimate from all patches that contain it.

**Predicting Matches from Noisy Observations.** The success of our match-averaging strategy in (2.2) depends on obtaining optimal values for the matching scores  $m_{ij}^g$ . Intuitively, we want  $m_{ij}^g$  to be high when the *clean* coefficients  $r_i^g$  and  $r_j^g$  are close, so that the averaging in (2.2) will attenuate noise and yield  $\hat{r}_i^g$  close to  $r_i^g$ . Conversely, we want  $m_{ij}^g$  to be low where the two sets of underlying clean coefficients are not similar, because averaging them would yield poor results, potentially worse than the noisy observation itself. However, note that while the optimal values of these matching scores depend on the characteristics of the clean coefficients  $\{r_i^g\}$ , we only have access to their noisy counterparts  $\{s_j^g\}$ .

Therefore, we train a neural network  $\mathcal{M}$  to predict the matching scores given a pair of larger noisy patches ( $16 \times 16$  in our implementation)  $y_i^+$  and  $y_j^+$  centered around  $y_i$  and  $y_j$  respectively:  $m_{ij} = \mathcal{M}(y_i^+, y_j^+)$ , where  $m_{ij} = [\dots, m_{ij}^g, \dots]$  is a vector of matching scores for all sets of coefficients. We don't require the output of the network  $\mathcal{M}$  to be symmetric ( $m_{ij}$  need not be the same as  $m_{ji}$ ), and we use the same network model for evaluating all patch pairs, being agnostic to their absolute or relative locations.

The matching network  $\mathcal{M}$  has a Siamese-like architecture as illustrated in Figure 2.3. It begins with a common feature extraction sub-network applied to both input patches to produce a feature-vector for each. This sub-network has a receptive field of  $16 \times 16$ , and includes a total of fourteen convolutional layers with skip connections [63] including at the final output (see Figure 2.3). The computed feature-vectors for each of the two inputs are then concatenated and passed through a comparison sub-network, which comprises of a set of

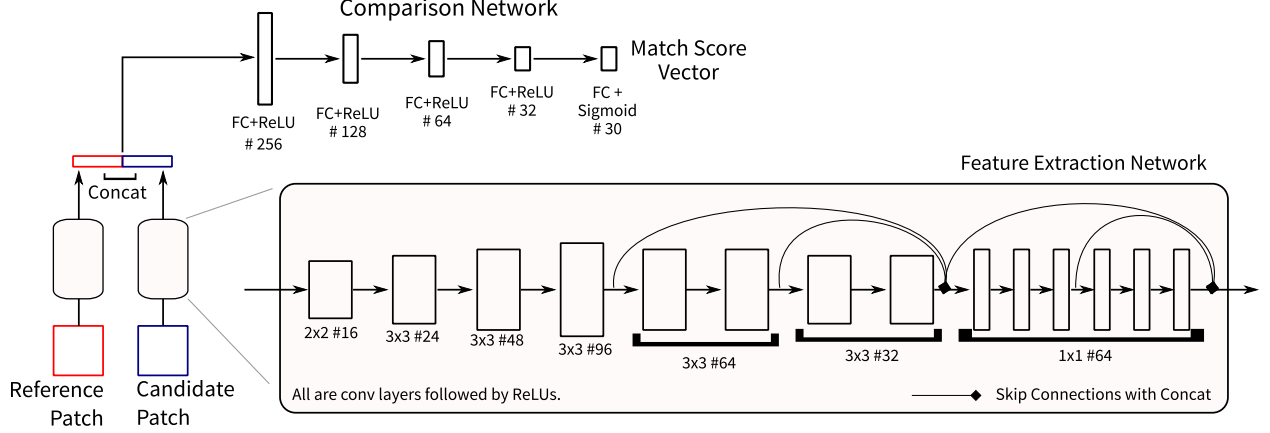


Figure 2.3: Matching Network Architecture. To produce the matching scores  $m_{ij}^g$ , we first extract a feature vector for all patches by passing the image through a feature extraction network, comprised of a set of convolutional layers with skip connections (where the join is performed by a concatenate operation). Then, for any pair of patches, we take the corresponding pair of feature vectors, and pass them after concatenation through a series of fully-connected layers. The final layer has a sigmoid activation, yielding scores that lie between 0 and 1.

five fully-connected layers. All layers have ReLU activations, except for the last which uses a sigmoid to produce the match-scores  $m_{ij}^g$ . These scores are thus constrained to lie in  $[0, 1]$ . Note that during inference, the feature extraction sub-network needs to be applied only once to compute feature-vectors for all patches in a fully-convolutional way. Only the final five fully connected layers need to be repeatedly applied for different patch pairs.

Observe that our matching network takes the noisy patches directly as input, while using the wavelet and color transforms to parameterize its outputs, as a means of providing a more fine-grained characterization of similarity between patches than a single score. Moreover, although the inputs to the network itself consist only of a pair of relatively small patches, it enables aggregation from a dense set of patches in a large neighborhood, through repeated application on multiple patch pairs and averaging as per (2.2).

**Training.** We train the matching network  $\mathcal{M}$  to produce matching scores that are optimal with respect to the quality of the denoised patches  $\hat{x}_i$ . Specifically, we use an  $L_2$  loss between the true and estimated clean patches:

$$L = \|x_i - \hat{x}_i\|^2 = \sum_g \|\hat{r}_i^g - r_i^g\|^2, \quad (2.3)$$

where the denoised coefficients  $\hat{r}_i^g$  are computed using (2.2) based on matching-scores predicted by the network. Note that the loss for a single patch  $x_i$  will depend on matching scores produced by the network for  $x_i$  paired with all candidate patches in its neighborhood  $\mathcal{N}_i$ .

While it is desirable to train the network in this end-to-end manner with our actual denoising approach, we empirically find that training the network with this loss from a random initialization often converges to a sub-optimal local minima. We hypothesize that this is because we compute gradients corresponding to a large number of matching scores (all candidates in  $\mathcal{N}_i$ ) with respect to supervision only from a single denoised patch.

Thus, we adopt a pre-training strategy using a loss defined on pairs of patches at a time, using a simplified loss for denoising patch  $i$  by averaging it with patch  $j$  as:

$$\hat{L}_{ij} = \sum_g \frac{\|s_i^g - r_i^g\|^2 + m_{ij}^g{}^2 \|s_j^g - r_i^g\|^2}{(1 + m_{ij}^g)^2}. \quad (2.4)$$

This is equivalent to the actual loss in (2.3) with performing the averaging in (2.2) with only one candidate patch  $j$ , by dropping the cross term between  $(x_i - y_i)$  and  $(x_i - y_j)$ , i.e., by assuming the noise is un-correlated with the difference between the two patches. It is interesting to note here if we assume that the deviations between reference and candidate patches are un-correlated, for different candidates  $j \in \mathcal{N}_i$ , then the optimal averaging weight for a given candidate is the same whether averaging with one or multiple candidates. The



modified loss in (2.4) serves as a good initial proxy for pre-training, but since the un-correlated deviation assumption does not hold in practice, we follow this with training with the actual loss in (2.3).

In particular, we pre-train the network for a number of iterations using the modified loss in (2.4)—constructing the training set by considering all non-overlapping patches  $i$  in an image, with random shuffling to select candidate  $j$  for each patch  $i$ , and train with respect to the loss of both matching  $i$  to  $j$  and vice-versa. This allows us to compute updates with respect to a much more diverse set of patches into a training batch, and to make maximal use of the feature extraction computation during training. The pre-training step is followed by training the network with the true loss in (2.3) till convergence—here, we extract a smaller number of training reference patches from each image, along with *all* their neighboring candidates.

**Final Estimates via Regression.** While the initial estimates produced by our method as described above are of reasonable quality, they are limited by (2.2) restricting every denoised patch to be a weighted average of observed noisy patches. To overcome this and achieve further improvements in quality, we use a second network trained via traditional regression to derive our final denoised estimate. Specifically, we adopt the architecture of IRCNN [185] which has seven dilated convolutional layers. In our case, this network takes a six-channel input formed by concatenating the original noisy input and our initial denoised estimate from match-based averaging. The output of the last layer is interpreted as a residual, and added to the initial estimates to yield the final denoised image output.

After the matching network has been trained, we generate sets of clean, noisy, and initial denoised estimates. This serves as the training set for this second network which is trained using an  $L_2$  regression loss. We find that this step leads to further improvement over our

	Method	$\sigma=50$			$\sigma=35$			$\sigma=25$		
		PSNR	SSIM	25%-ile	PSNR	SSIM	25%-ile	PSNR	SSIM	25%-ile
Urban-100	*CBM3D [32]	27.94	0.843	25.96	29.27	0.875	27.07	31.38	0.912	29.20
	IRCNN [185]	27.69	0.842	25.63	29.50	0.881	27.39	31.20	0.911	29.06
	FFDNet [186]	28.05	0.850	25.93	29.78	0.887	27.61	31.40	0.914	29.17
	Ours-Blind	28.57	0.859	26.47	30.21	0.893	28.04	31.70	0.917	29.49
	Ours	<b>28.62</b>	<b>0.862</b>	<b>26.52</b>	<b>30.26</b>	<b>0.895</b>	<b>28.09</b>	<b>31.81</b>	<b>0.919</b>	<b>29.59</b>
Kodak-24	*CBM3D [32]	28.45	0.775	26.51	29.90	0.821	27.77	31.67	0.868	29.51
	IRCNN [185]	28.81	0.792	26.76	30.43	0.838	28.32	32.03	0.878	29.91
	CDnCNN [184]	28.85	-	-	30.46	-	-	32.03	-	-
	FFDNet [186]	28.98	0.793	26.89	30.57	0.841	28.41	32.13	0.879	29.95
	Ours-Blind	29.21	0.803	27.11	30.78	0.849	28.63	32.31	<b>0.884</b>	30.14
	Ours	<b>29.25</b>	<b>0.805</b>	<b>27.15</b>	<b>30.81</b>	<b>0.849</b>	<b>28.66</b>	<b>32.34</b>	<b>0.884</b>	<b>30.19</b>
CBSD-68	*CBM3D [32]	27.38	0.767	25.07	28.89	0.821	26.46	30.71	0.872	28.25
	*CBM3D-Net [176]	27.48	-	-	-	-	-	30.91	-	-
	*CNL-Net [86]	27.64	-	-	-	-	-	30.96	-	-
	IRCNN [185]	27.86	0.792	25.54	29.50	0.844	27.14	31.16	0.886	28.81
	CDnCNN [184]	27.92	-	-	29.58	-	-	31.23	-	-
	FFDNet [186]	27.96	0.792	25.56	29.58	0.844	27.14	31.21	0.886	28.78
	Ours-Blind	28.03	0.797	25.65	29.62	0.848	27.21	31.22	<b>0.888</b>	28.82
	Ours	<b>28.06</b>	<b>0.799</b>	<b>25.69</b>	<b>29.64</b>	<b>0.849</b>	<b>27.24</b>	<b>31.24</b>	<b>0.888</b>	<b>28.85</b>
McMaster	*CBM3D [32]	28.52	0.794	26.41	29.92	0.833	27.73	31.66	0.874	29.49
	IRCNN [185]	28.91	0.807	26.78	30.59	0.851	28.48	32.18	0.885	30.11
	CDnCNN [184]	28.61	-	-	30.14	-	-	31.51	-	-
	FFDNet [186]	29.18	0.816	26.99	30.81	0.857	28.62	<b>32.35</b>	0.889	<b>30.20</b>
	Ours-Blind	29.31	0.824	27.14	30.85	0.861	28.69	32.31	<b>0.890</b>	30.14
	Ours	<b>29.35</b>	<b>0.826</b>	<b>27.16</b>	<b>30.90</b>	<b>0.863</b>	<b>28.70</b>	32.33	<b>0.890</b>	30.17

\*Other methods that are based on internal image statistics.

Table 2.1: Denoising Performance at Various Noise Levels on Different Datasets. We report performance in terms of Average PSNR (dB) and SSIM. To gauge robustness, we also report the 25<sup>th</sup>-ile worst-case PSNR (dB), computed across  $8 \times 8$  patches across each dataset. Ours-Blind refers to results from a common model of our method that is trained for a range of noise levels  $\sigma \in [0, 55]$  (and does not have knowledge of the specific noise level of its input).

initial estimates, while also outperforming state-of-the-art denoising networks (including IRCNN [185] itself).

## 2.2.3 Experiments

**Preliminaries.** We train our algorithm on a set of 1600 color images from the Waterloo exploration dataset [112], and 168 images from the BSD-300 [116] train set, using the

remaining 32 images for validation and parameter setting. We train our network using noisy observations generated by adding Gaussian noise to clean images in the training set. Unless otherwise specified, we construct the candidate set  $\mathcal{N}_i$  of patches by considering all the overlapping patches in a  $31 \times 31$  search window around patch  $i$ .

We use Adam [71] to train both the matching and regression networks, with an initial learning rate of  $10^{-3}$ . We pre-train the matching network for a 100k iterations based on the modified loss (2.4), with batches of 16 images and pairing all non-overlapping patches with randomly shuffled counterparts. This leads to a large number of ordered matching pairs for pre-training in each batch. We then continue training the matching network with the true loss in (2.3), in this case forming a batch with 256 unique reference patches from various images, and computing matching scores for each with respect to all  $31^2$  candidates. We train with this loss till saturation, with two learning rate drops of  $10^{0.5}$ . Once the matching network is trained, we store a set of noisy and denoised version of our training set, and use these to train the regression network (with the same training schedule, but without pre-training).

**Denoising with Known Noise Level.** We evaluate our method for the task of color image denoising at five different noise levels, corresponding to additive white Gaussian noise with standard deviations of  $\sigma = 25, 35$  and  $50$  gray levels. We train a separate network model for each level, and report their performance in Table 2.1 on four datasets: Urban-100 [64], Kodak-24 [47], CBSD-68 [137], and McMaster [187]. For comparison, we also show results from a number of other state-of-the-art color denoising methods [32, 86, 176, 184, 185, 186]. We evaluate performance in terms of the standard PSNR and SSIM [164] metrics, and to measure robustness, report worst-case errors as the 25<sup>th</sup>%-ile among PSNR values of all individual  $8 \times 8$  patches in all images in each dataset. We find that our results are consistently more accurate across all datasets, with significant improvements over state-of-the-art methods

at higher noise levels. Moreover, not only are our denoised estimates more accurate on average in terms of PSNR and SSIM, our worst-case performance is also better—highlighting the robustness of our approach.

We include examples of denoised images in Figure 2.4 for a qualitative evaluation, and see that denoised results from our method often contain better reconstructions of texture and image detail than state-of-the-art denoising methods. In general, we find that our method has an advantage when a scene contains many repeating textures as expected, and also when it contains unique patterns—that are rare in training data and which regression-based methods are thus unable to reliably estimate. For images with limited repeating patterns, the burden of denoising then falls more to our second regression network, which then is able to still achieve results of acceptable quality at the level of standard regression-based methods.

**Blind Denoising.** Next, we consider the task of blind denoising, when the level of Gaussian noise in an observed image is unknown. For this, we follow the approach of [184] in training a common model for a range of noise levels  $\sigma \in [0, 55]$ , by adding Gaussian noise with  $\sigma$  chosen randomly for each image during training. Note that unlike for FFDNet [186], the noise level for a specific input image is *not* provided to our model. Table 2.1 also includes an evaluation of this version of our method (as Ours-Blind). We find that its performance is only slightly lower than that of our noise-specific networks, and still better in almost all cases than that of state-of-the-art methods that are aware of the level of noise in their inputs. This represents an attractive and practically useful variant of our method—which does not require maintaining multiple models for each noise level, and can be applied even when the noise level is unknown.

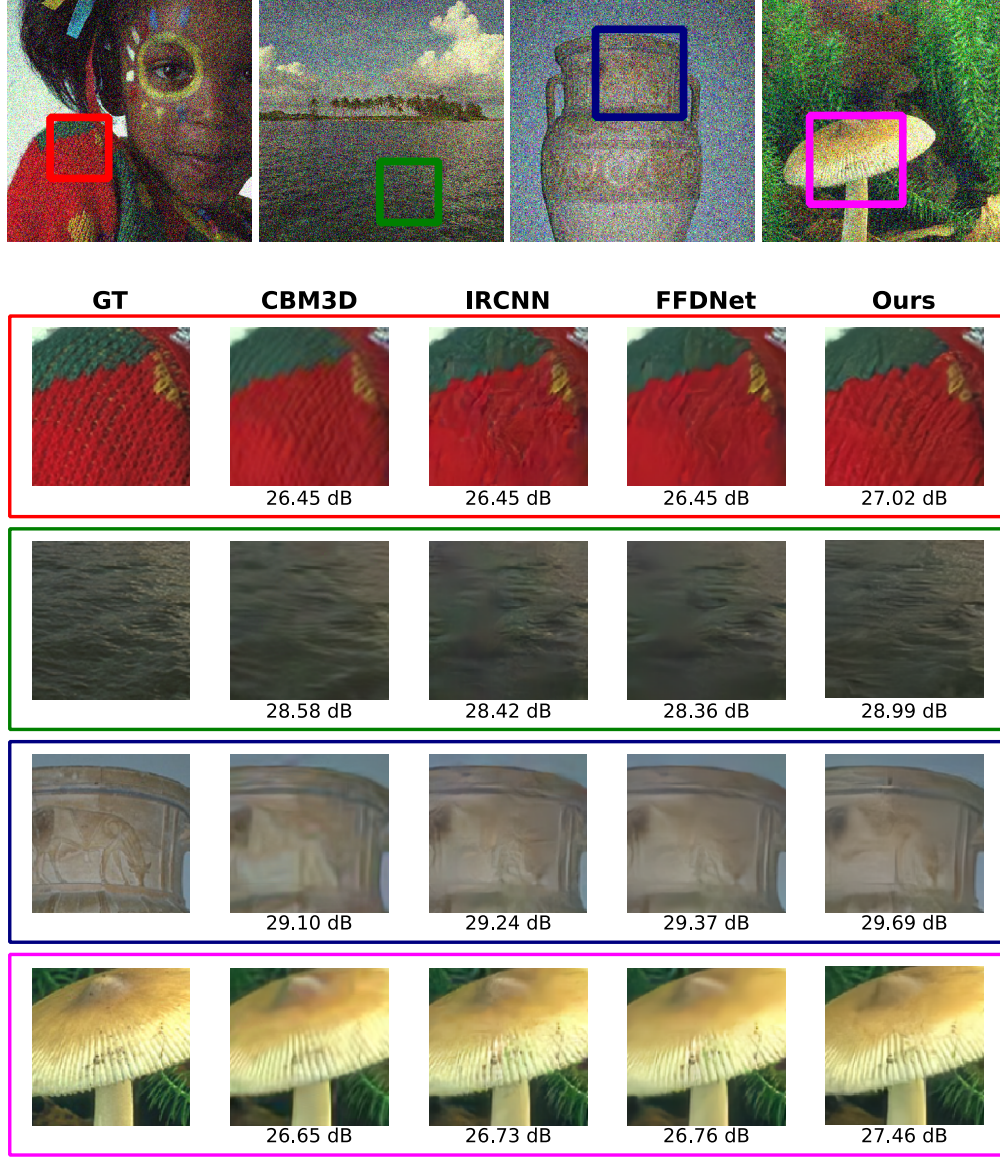


Figure 2.4: Example Crops of Denoised Images ( $\sigma = 50$ ). Compared to state-of-the-art denoising algorithms (IRCNN [185], DnCNN [184], and FFDNet [186]), we see that our overall method is often able to recover texture and detail with higher fidelity, by exploiting similar patterns in the input image itself.

**Training with Limited Data.** For many forms of image data and measurement models (*e.g.*, medical images), a large amount of training data is difficult to acquire. Here, our method presents an advantage because of its focus on leveraging common patterns and textures in the input image itself, rather than those it has observed previously in a training

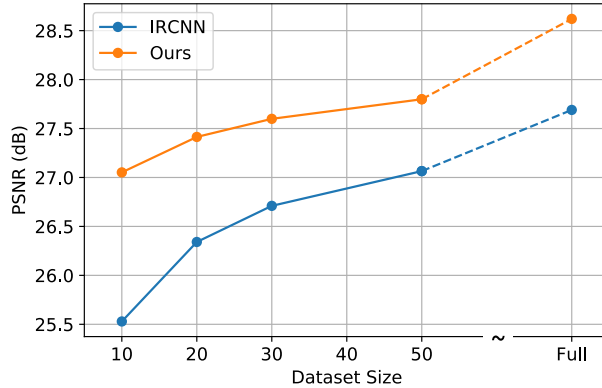


Figure 2.5: Effect of training set size. We report average PSNR on Urban-100 [64] for denoising at  $\sigma = 50$  with our method and IRCNN [185], when both are trained with a limited number of training images (“Full” represents using the entire training set for our method, and the official model for IRCNN). While our estimates are always more accurate than those from IRCNN, the gap is especially higher when the number of training images is small.

set. We demonstrate this advantage by comparing our method to IRCNN [185] in Figure 2.5, when both methods are trained with only a few training images (selected from our complete training set).

We assume a fixed known noise level of  $\sigma = 50$ , and train versions of both models with different training set sizes—ranging from 10 to 50 images. Since overfitting is an issue with so little data, we track the performance of both methods on a validation set of 32 images through training, and choose the version with highest validation accuracy. We do not drop the learning rate for either method in this setting. Figure 2.5 shows the average PSNR for both methods on the Urban-100 dataset [64], for different training set sizes. While our method outperforms IRCNN [185] in all cases, the performance gap is notably larger for smaller training sets—at 1.5 dB when training with only 10 images.

**Matching Network Analysis.** Our matching network is a key component of our denoising algorithm. We end by analyzing its performance when applied to neighborhoods of different

Window Size	15	23	31	31 (No Pre-training)
PSNR (dB)	31.31	31.40	31.46	31.35
Run Time	1.07s	2.47s	4.42s	4.42s

Table 2.2: Window Size and Pre-Training Ablation. We report average PSNR (db) of the initial match-averaged estimates from our method on a validation set for  $\sigma = 25$ . Run-times are for  $256 \times 256$  images on a 1080Ti GPU.

sizes, and the role that pre-training plays in convergence to a good solution. We also visualize the matching scores it generates, and how these differ across different groups of sub-bands.

In Table 2.2, we characterize the trade-off between quality and computational cost when choosing different search window sizes over which to match and average patches. For different window sizes, we report average PSNR (over our validation set) for our initial match-averaged estimates when training with a known noise level of  $\sigma = 25$ . We also report the corresponding running time required to compute matching scores and perform the averaging for different window sizes—for a  $256 \times 256$  input image on an NVIDIA 1080Ti GPU. Note that computing the initial estimates takes a majority of the time in our denoising method—the following regression step takes only an additional 0.01 seconds, and is independent of window size.

As expected, running time goes up roughly linearly with the number of candidate matches (i.e., as square of the search window size), but we find that the drop in PSNR is a modest 0.06 dB when going down to a  $23 \times 23$  window. Table 2.2 also demonstrates the importance of pre-training, and reports performance (again, of our initial estimates) achieved by a network that is initialized with random weights instead of with pre-training. We find that this leads to a PSNR drop of about 0.1 dB, highlighting that pre-training is important for convergence to a good model.

For a number of reference patches cropped from different training images, and their corresponding search windows, we visualize the matching scores predicted by our network in



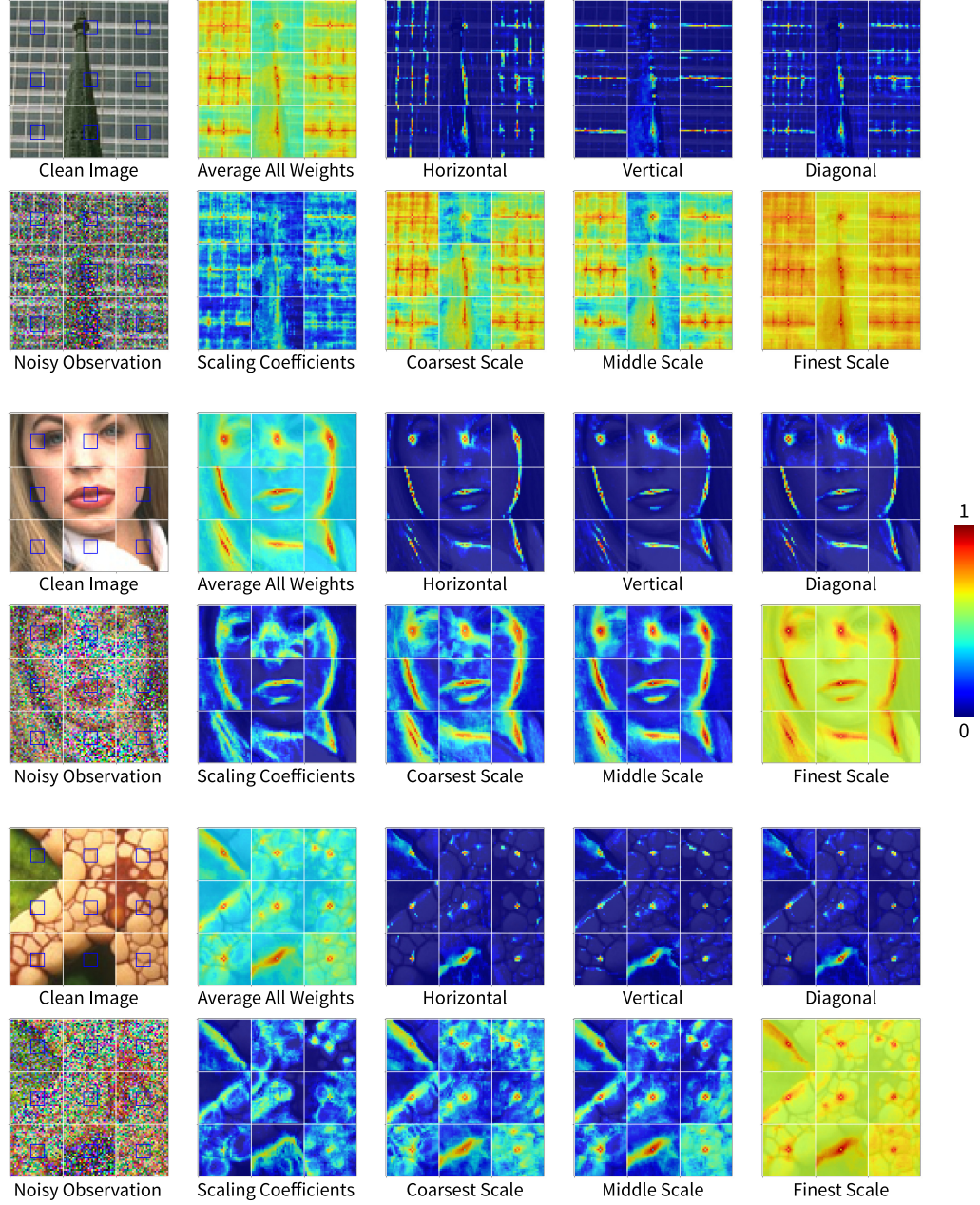


Figure 2.6: Visualization of Matching Score Distributions in Different Sub-bands. We show reference patches (indicated by blue squares) along with their local search windows from various images of the training set (9 windows per image), and visualize the matching scores predicted by our network. We show the predicted weights averaged across all sub-bands, as well as specific to different scales (averaging over color and orientation at each scale), and orientations (averaging over color and scale).



Figure 2.6. We show the average matching score across all sub-bands, as well as average weights corresponding to combinations of sets at the same wavelet scale (averaging over color channels and orientation), and at the same orientation (averaging over scale and color). We see that the matching network produces very different averaging weights for different sub-bands.

We find that the weights tend to be generally higher at the finest scale (indicating more averaging), and lowest for the scaling coefficients. This is likely because the highest-frequencies are close to zero in most patches, and thus to each other. For lower-frequencies and DC values, the network selects only those patches that are close to the reference patch (in the clean image). For different orientations, the high matches are sometimes concentrated at different locations for the same reference, especially when there are strong edges and repeating textures. Thus, free from the constraint of matching patches as a whole with a single score, our network finds different sets of matches for different sub-bands in order to achieve optimal denoising.

#### **2.2.4 Discussion**

In this section, we proposed to employ neural networks to identify and exploit recurring patterns in natural images for single image denoising. Our network provided a fine-grained characterization of similarity, in terms of separate scores for different corresponding sub-band components, and thus enabled the recovery of high-quality denoised estimates. We also showed that our network is especially useful in regimes where training data is scarce, being able to achieve relatively higher performance from training on a small number of examples than standard regression-based methods. A natural direction of future work lies in exploring applications of our approach, of characterizing sub-band level self-similarity, to other image-like signals such as depth maps and motion-fields.

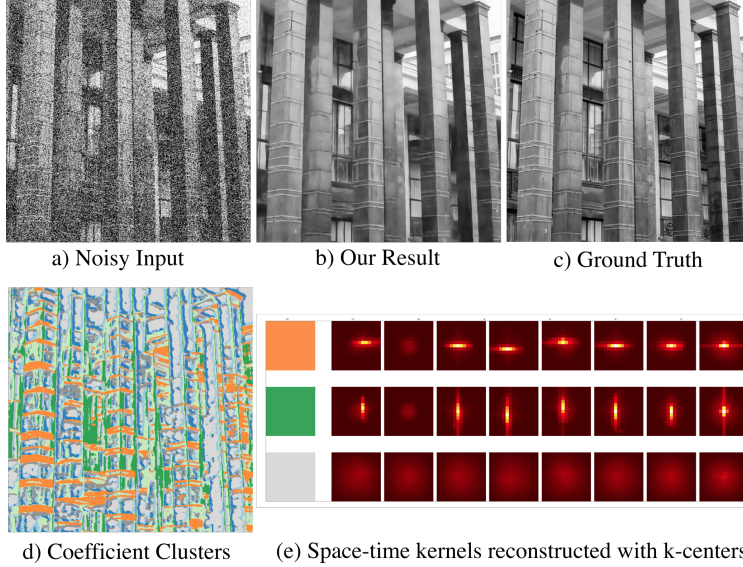


Figure 2.7: (*top*) Qualitative result of our denoising network. The proposed approach (b) recovers fine geometric details from a very noisy image burst (a, only one frame is shown). (*Bottom*) our per-burst bases can compactly represent a large variety of kernels by exploiting the redundancy of local image structures. We clustered the per-pixel coefficients predicted from the noisy burst using  $k$ -means. The clusters we obtain show strong spatial structure (d). For instance, we can identify a cluster corresponding to horizontal image edges (e, orange), one corresponding to vertical edges (e, green), and another for homogeneous regions with no structure (e, gray).

## 2.3 Exploiting self-similarity in denoising kernels for burst photography

### 2.3.1 Introduction

Despite the impressive results of single image denoising, the physical limits imposed by read noise and photon noise, especially for hand-held smartphone cameras with small apertures and sensors, make capturing a single high-quality image under low-light environments challenging. To this end, burst photography [57, 109, 120] have been developed to enable high-quality photography in such conditions and are increasingly being deployed in commercial mobile

cameras [57, 99]. A burst captures a sequence of short-exposure frames of the scene that are free of motion-blur, but with a high amount of noise in each frame and relative motion between frames. By accounting for this relative motion and using the fact that the noise is independent across frames, burst denoising attempts to aggregate these inputs and predict a single noise- and blur-free image estimate.

While classical approaches seek to explicitly estimating inter-frame motion [57, 58, 59, 74, 109], to align frames and then denoise, Mildenhall *et al.* [120] proposed to combine these two steps with a neural network. Rather than explicitly estimating inter-frame motion [57, 58, 59, 74, 109], their method produces denoised estimates at each pixel as a weighted average of observed noisy intensities in a window around that pixel’s location in all frames. These averaging weights, or kernels, are allowed to vary from pixel-to-pixel to implicitly account for motion and image discontinuities, and are predicted from the noisy input burst using a “kernel prediction network” (KPN).

However, KPNs need to produce an output that is significantly higher-dimensional than the denoised image—even for  $5 \times 5$  kernels with eight frames, a KPN must predict 400 times as many kernel weights as image intensities. This comes with significant memory and computational costs, as well as difficulty in training given the many degrees of freedom in the output. As a result, KPNs have so far been used only with small kernels. This limits their denoising ability by preventing averaging across bigger spatial regions, and over frames with larger relative motion.

In this section, we take the recourse to self-similarity and describe a neural approach that predict large denoising kernels, and thus benefit from wider aggregation, while simultaneously limiting output dimensionality at each pixel, making the prediction network easier to train and compute- and memory-efficient. Self-similarity is particularly strong for burst photography

because we expect both *spatial* structure in the form of similar patterns that recur within a frame and across frames of the same scene, and *temporal* structure caused by consistency in scene and camera motion. Given the expected self-similarity and structure in the image intensities themselves, we argue that the corresponding denoising kernels must also have similar structure. Specifically, while allowing for individual per-pixel denoising kernels to be large, we assume that all the kernels for a given image span a lower-dimensional subspace.

Based on this observation, we train a network that, given an input noisy burst, predicts both a global low-dimensional basis set of large kernels, and per-pixel coefficient vectors relative to this basis. This corresponds to a significantly lower-dimensional output than a KPN that predicts arbitrary kernels of the same size at each pixel. Enforcing this structure on the denoising kernels acts as a form of regularization, and leads to state-of-the-art denoising performance with significantly higher-quality ( $>1$  dB PSNR) than regular KPNs. Beyond reducing memory usage and computational burden, the structure of our output enables the final kernel filtering step to be performed much more efficiently in the Fourier domain. We show this in terms of required number of FLOPs, as well as experimentally with actual run-times.

### 2.3.2 Method

In burst denoising, we are given an input noisy burst of images  $I[n, t]$ , where  $n$  indexes spatial locations and  $t \in \{1, \dots, T\}$  the different frames in the burst. Using a heteroscedastic Gaussian noise model [46], which accounts for both read and shot noise, we relate this to the corresponding noise-free frames  $R[n, t]$  as:

$$I[n, t] \sim \mathcal{N}(R[n, t], \sigma_r^2 + \sigma_s^2 R[n, t]), \quad (2.5)$$

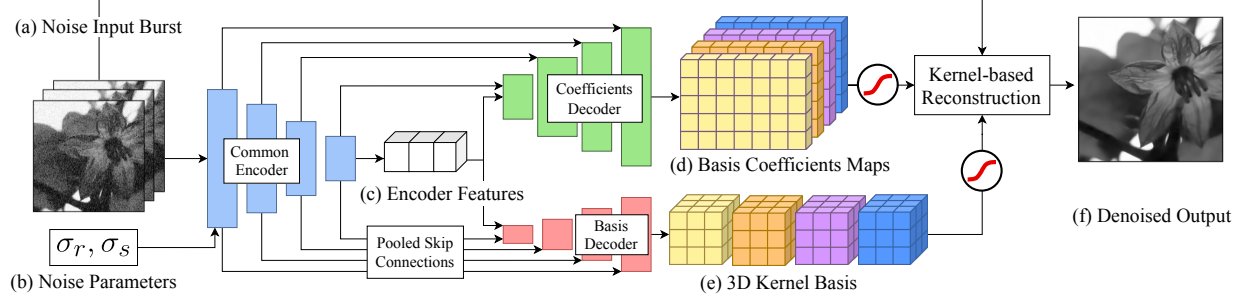


Figure 2.8: Our basis prediction network takes as input a burst of noisy input frames (a) together with the noise parameters (b). The frames are encoded into a shared feature space (c). These features are then decoded by two decoders with skip connections into a burst-specific basis of 3D kernels (e) and a set of per-pixel mixing coefficients (d). Both the coefficients and basis kernels are individually unit-normalized. Finally, we obtain per-pixel kernels by mixing the basis elements according to the coefficients and we apply them to the input burst to produce the final denoised image (f).

where  $\sigma_r^2$  and  $\sigma_s^2$  are the read- and shot-noise parameters. Choosing the first frame as reference, our goal is to produce a single denoised image  $\hat{R}[n]$  as an estimate of the first noise-free frame  $R[n, 1]$ .

### 2.3.3 Kernel-based burst denoising

Rather than train a network to regress  $\hat{R}$  directly, Kernel Prediction Networks output a field of denoising kernels  $w_n[\delta, t]$ , one for each pixel  $n$  at each frame  $t$ . The kernels have a spatial support  $K \times K$ , indexed by  $\delta$ , with separate weights for each frame. Given these predicted kernels, the denoised estimate  $\hat{R}$  is formed as:

$$\hat{R}[n] = \sum_t \sum_{\delta} w_n[\delta, t] I[n - \delta, t]. \quad (2.6)$$

A key bottleneck in this pipeline is the prediction of this dense kernel field  $w$ , which requires producing  $K^2T$  numbers at *every* pixel of the output. Since networks with high-dimensional

outputs are both expensive and require learning a large number of parameters in their last layer, KPNs have typically been used only with small kernels ( $K = 5$  in [120]).

**Basis Prediction Networks.** Instead of directly predicting unconstrained kernels for each spatial location, we designed a network that outputs: (1) a global *kernel basis*  $v_b[\delta, t]$ , of size  $K^2T \times B$  with  $b \in \{1, \dots, B\}$ ; and (2) a  $B$  dimensional coefficient vector  $c_n[b]$  at each spatial location.

$$w_n[\delta, t] = \sum_b v_b[\delta, t] c_n[b]. \quad (2.7)$$

Note that we typically choose the number of basis kernels  $B \ll K^2T$ . This implies that all the kernels for a given burst lie in a low-dimensional subspace, but this subspace will be different for different bursts, *i.e.* the basis is burst-specific. This procedure allows us to recreate a full kernel field with far fewer predictions. Assuming a  $W \times H$  resolution image, we need only make  $WHB + K^2TB$  predictions to effectively recreate a kernel field of size  $WHK^2T$ .

We designed our network following an encoder-decoder architecture with skip connections [136]. Our model, however, has two decoder branches, one for the basis, the other for the coefficients (Figure 2.8). The encoder is shared between the two branches because the meaning of the coefficients  $c$  is dependent on the predicted basis  $v$  in Equation (2.7), so the two outputs need to be co-ordinated. This encoder takes the noisy burst and noisy parameters as input, and through multiple levels of downsampling and global average pooling at the end, yields a single global feature vector as its encoding of the image. The per-pixel coefficients  $c$  are then decoded from the encoder bottleneck to the full image resolution  $W \times H$ , with  $B$  channels as output. The common basis  $v$  is decoded up to *distinct* spatial dimensions — that of the kernels  $K \times K$  — with  $B \times T$  output channels.

Since the basis branch decodes to a different spatial resolution, we need a careful treatment of the skip connections. Unlike a usual U-Net, the encoder and decoder feature size do not match. Specifically, a pixel  $\delta$  in the basis kernel  $v_b[\delta, \cdot]$  has no meaningful relation to a pixel  $n$  in the input frames  $I[n, \cdot]$ . Therefore, in the skip connections from the shared encoder to the basis decoder, we apply a global spatial average pooling of the encoder’s activations, and replicate the average vector to the resolution of the decoder layer. This mechanism ensures the encoder information is globally aggregated without creating nonsensical correspondences between kernel and image locations, while allowing features at multiple scales of the encoder to inform the basis decoder.

We ensure each of the reconstructed kernel  $w$  has positive weights that sum to one, to represent averaging. We implement this constraint using soft-max normalizations on *both* the coefficient and basis decoder outputs. So every 3D kernel of the basis  $v_b[\cdot, \cdot]$  and every coefficient vector  $c_n[\cdot]$  is normalized individually.

Our network is trained with respect to the quality of the final denoised output  $\hat{R}$ —with an  $L2$  loss on intensities and  $L1$  loss on gradients. Like [120], we additionally use a per-frame loss to bias the network away from relying only on the reference frame. We do this with separate losses on denoised estimates from each individual frame of the input burst (formed as  $\hat{R}_t[n] = T \sum_{\delta} w_n[\delta, t] I[n - \delta, t]$ ). These are added to main training loss, with a weight that is decayed across training iterations.

**Efficient Fourier domain filtering.** Filtering by convolution with large kernels is commonly implemented in the Fourier domain, where the filtering complexity is quasilinear in image size, while the complexity of direct convolution scales with the product of image- and kernel-size. But because the kernels  $w$  in KPNs vary spatially, Equation (2.6) does not represent a standard convolution, ruling out this acceleration.

In our case, because our kernels are defined with respect to a small set of “global” basis vectors, we can leverage Fourier-domain convolution to speed up filtering. We achieve this by combining and re-writing the expressions in Eq. (2.6) and Eq. (2.7) as:

$$\begin{aligned}
\hat{R}[n] &= \sum_t \sum_{\delta} w_n[\delta, t] I[n - \delta, t] \\
&= \sum_t \sum_{\delta} \sum_b v_b[\delta, t] c_n[b] I[n - \delta, t] \\
&= \sum_b c_n[b] \sum_t \sum_{\delta} v_b[\delta, t] I[n - \delta, t] \\
&= \sum_b c_n[b] \sum_t (I[\cdot, t] \star v_b[\cdot, t]) [n],
\end{aligned} \tag{2.8}$$

where  $\star$  denotes standard spatial 2D convolution with a spatially-uniform kernel.

In other words, we first form a set of  $B$  filtered versions of the input burst  $I$  by standard convolution with each of the basis kernels—convolving each frame in the burst  $I$  with the corresponding “slice” of the basis kernel—and then taking a spatially-varying linear combination of the filtered intensities at each pixel based on the coefficients  $c$ . We can carry out these standard convolutions in the Fourier domain as:

$$I[\cdot, t] \star v_b[\cdot, t] = \mathcal{F}^{-1} \left( \mathcal{F}(I[\cdot, t]) \cdot \mathcal{F}(v_b[\cdot, t]) \right), \tag{2.9}$$

where  $\mathcal{F}(\cdot)$  and  $\mathcal{F}^{-1}(\cdot)$  are spatial forward and inverse Fourier transforms. This is significantly more efficient for larger kernels, especially since we need not repeat forward Fourier transform of the inputs  $I$  for different basis kernels.



Method	Gain $\propto 1$	Gain $\propto 2$	Gain $\propto 4$	Gain $\propto 8$
HDR+ [57]	31.96	28.25	24.25	20.05
BM3D [32]	33.89	31.17	28.53	25.92
NLM [16]	33.23	30.46	27.43	23.86
VBM4D [114]	34.60	31.89	29.20	26.52
Direct	35.93	33.36	30.70	27.97
KPN [120]	36.47	33.93	31.19	27.97
KPN* ( $K = 5$ )	36.35	33.69	31.02	28.16
MKPN* [115]	36.88	34.22	31.45	28.52
BPN (ours)	<b>38.18</b>	<b>35.42</b>	<b>32.54</b>	<b>29.45</b>

Table 2.3: Denoising performance on a synthetic grayscale benchmark [120]. We report performance in terms of Average PSNR (dB). Following [115, 120], our BPN was not trained on the noise levels implied by the gain in the fourth column. Numbers for KPN\* ( $K = 5$ ) and MKPN\* are based on our implementation of these techniques. Numbers for all other methods, including the end-to-end regression model to directly synthesize denoised pixel intensities (denoted as Direct), are from [120]. Our method widely outperforms all prior methods at all noise levels.

### 2.3.4 Experiments

We closely follow Mildenhall et al. [120] for training and evaluation. Our model is designed for bursts of  $T = 8$  frames with resolution  $128 \times 128$ . Following the procedure of [120], we use training and validation sets constructed from the Open Images dataset [76], with shot and read noise parameters uniformly sampled in the log-domain:  $\log(\sigma_r) \in [-3, -1.5]$ , and  $\log(\sigma_s) \in [-4, -2]$ . We also use [120]’s set of 73 grayscale test images for evaluation.

Our default configuration uses bases with  $B = 90$  kernels of size  $K = 15$ . We train our network (as well as all ablation baselines) using Adam [72] with a batch-size of 24 images, and an initial learning rate of  $10^{-4}$ . We train for a total of about 600k iterations, dropping the learning twice, by  $\sqrt{10}$  each time, whenever the validation loss saturates.

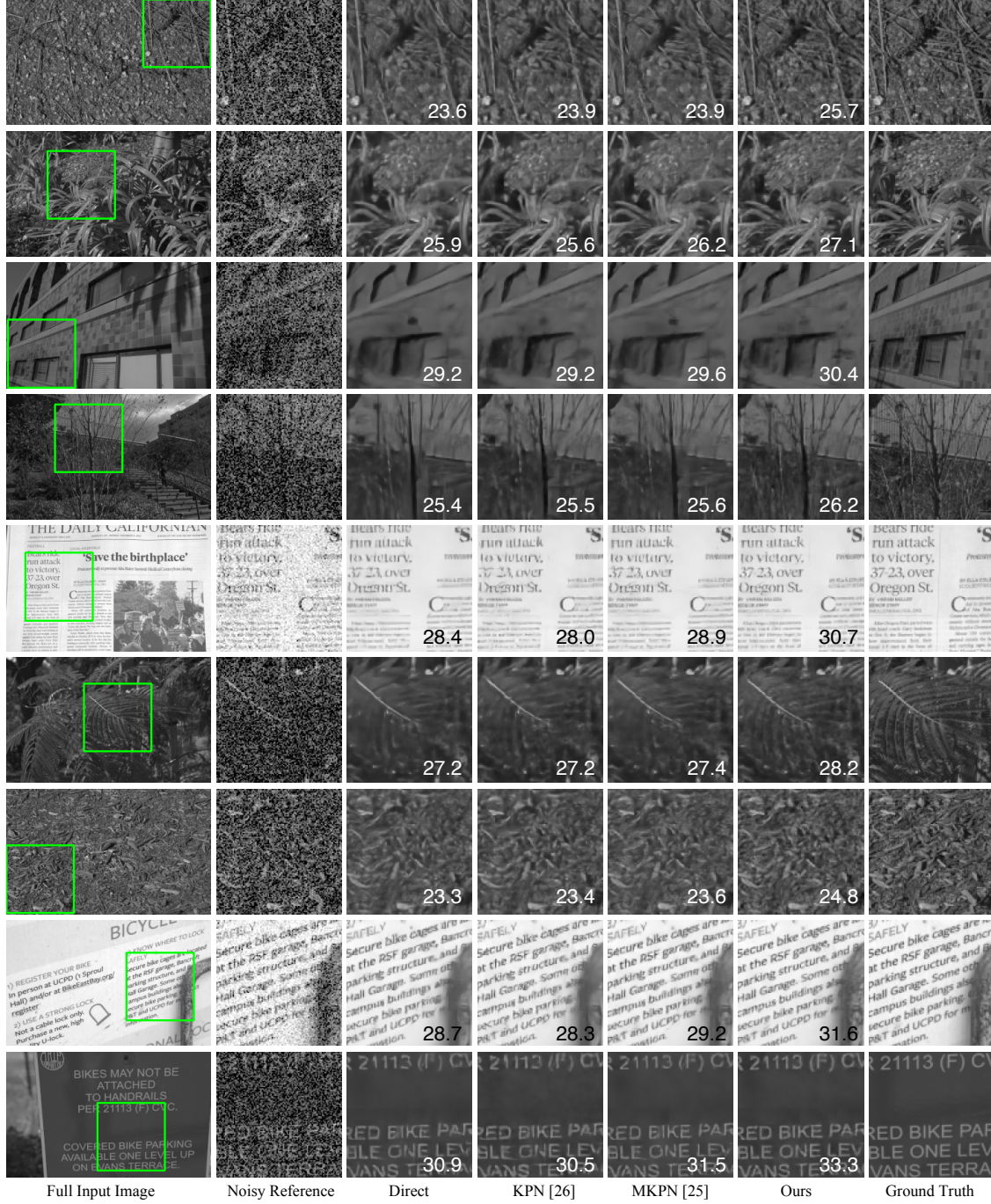


Figure 2.9: We illustrate denoising performance on a benchmark synthetic grayscale test set [120] for our method, a direct prediction network (which directly regresses denoised pixels), and two KPN variants [115, 120] with the same kernel size  $K = 15$  as our method. The numbers in inset refer to the PSNR (dB) on the full image. In addition to better quantitative performance, our method does better at reproducing perceptual details like textures, edges, and text.

**Denoising performance.** Table 2.3 reports the PSNR of our denoised outputs on the grayscale test set [120]. Each noise level corresponds to a sensor gain value (one stop increments of the ISO setting in a camera). The gains correspond to the following values for  $(\log(\sigma_s), \log(\sigma_r))$ :  $1 \rightarrow (-2.2, -2.6)$ ,  $2 \rightarrow (-1.8, -2.2)$ ,  $4 \rightarrow (-1.4, -1.8)$ ,  $8 \rightarrow (-1.1, -1.5)$ . The highest noise level, denoted as  $\text{Gain} \propto 8$ , lies outside the range we trained on. We use it to evaluate our model’s extrapolation capability. In addition to our own model, we also report results for a motion-alignment-based method [57], several approaches based on non-local filtering [16, 32, 114], as well as the standard KPN burst denoiser [120]—which is the current state-of-the-art. Since we did not have access to the original KPN model, we implemented a version ourselves (that we use in ablations in the next section) and also report its performance in Table 2.3. We find it closely matches those from [120]). Additionally, we train a network to directly regress the denoised pixel values from the input burst (*i.e.*, without kernels), as well as our implementation of [115] with a larger kernel size of  $K = 15$  for fair comparison.

We find that our method outperforms KPN [120] by a significant margin, over 1 dB PSNR at all noise levels. Our implementation of [115] also does well, but remains inferior to our model. We show qualitative results for a subset of methods in Figure 2.9. Our have fewer artifacts, especially in textured regions and around thin structures like printed text.

**Ablation and analysis.** Our approach leads to better denoising quality because it enables larger kernels without vastly increasing the network’s output dimensionality and number of learnable parameters. To tease apart the contributions of kernel size and the structure of our kernel decomposition, we conduct an ablation study on our validation set. The results can be found in Table 2.4.

The performance gap between the test and validation set results (Table 2.3 and 2.4) comes from differences in the datasets themselves.

	Gain $\propto 1$	Gain $\propto 2$	Gain $\propto 4$	Gain $\propto 8$
KPN ( $K = 15$ )	34.29	31.80	28.23	24.86
Separable ( $K = 15$ )	34.67	32.05	28.52	25.12
Ours ( $K = 5$ )	35.70	33.02	29.16	25.57
Ours ( $K = 9$ )	36.22	33.41	29.56	25.94
Ours ( $B = 10$ )	35.31	32.69	28.95	25.43
Ours ( $B = 50$ )	36.10	33.33	29.45	25.88
Ours ( $B = 130$ )	36.27	33.47	29.57	<b>25.99</b>
<b>Ours (<math>K = 15, B = 90</math>)</b>	<b>36.29</b>	<b>33.57</b>	<b>29.62</b>	<b>25.99</b>
Common Spatial Basis	35.71	33.04	29.23	25.71
Per-frame Spatial Basis	36.21	33.46	29.56	25.92
Fixed basis	34.66	32.15	28.68	25.39

Table 2.4: Ablation study on our validation dataset. Performance is reported in terms of Average PSNR (dB). Beyond motivating our parameter choices ( $K = 15, B = 90$ ), this demonstrates that our use of a burst-specific spatio-temporal basis outperforms standard KPN [120], separable spatial kernels, a common spatial basis for all burst frames, separate spatial bases per-frame, and a fixed, input-agnostic basis. All these variants were trained with the same settings ( $K = 15, B = 90$ ) as our model.

*Kernel Size.* As a baseline, we consider using KPN directly with our larger kernel size of  $K = 15$ . We also consider predicting a single separable kernel at that size ([115] predicts separable kernels at multiple sizes, and adds them together). We find that our network outperforms the large kernel KPN variant at all noise levels—suggesting that simply increasing the kernel size is not enough. It also outperforms separable kernel prediction, suggesting that a low-dimensional subspace constraint better captures the structure of natural images than spatial separability.

For completeness, we also evaluate our basis prediction network with smaller kernels,  $K = 9$  and  $K = 5$ . Although, this leads to a drop in performance compared to our default configuration, these variants still perform better than the original KPN—suggesting our approach has a regularizing effect that benefits even smaller kernels.

*Basis Size.* The number of basis elements in our default configuration,  $B = 90$ , was selected from a parameter search on the validation set. We include this analysis in Table 2.4, reporting PSNR values for  $B$  ranging from 10 to 130. We find that bases with fewer than 90 kernels lead to a drop in quality. The larger bases,  $B = 130$ , also performs very slightly worse than  $B = 90$ . We hypothesize that large bases start to have too many degrees of freedom. This increases the dimensionality of the network’s output, which negates the benefits of a subspace restriction.

*Spatial vs. Spatio-temporal Basis Decomposition.* Note that we define our basis as a subspace to span 3D kernels—i.e., each of our basis elements  $v_b$  is a 3D spatio-temporal kernel. We predict a single weight  $c_n[b]$  at each location, which is applied to corresponding spatial kernels  $v_n[\cdot, t]$  for all frames  $t$ . However, there are other possible choices for decomposing 3D kernels, and we consider two of these in our ablation (Table 2.4). In both cases, we output coefficients  $c_{n,t}[b]$  that vary per-frame, in addition to per-location—and are interpreted as separate coefficients corresponding to a spatial basis kernel. In one case, we use a *common spatial basis*  $v_b[\delta]$  across all frames, with  $w_n[\delta, t] = \sum_b c_{n,t}[b]v_b[\delta]$ . In the other, we have a *per-frame spatial basis*  $v_{b,t}[\delta]$  for each frame, and  $w_n[\delta, t] = \sum_b c_{n,t}[b]v_{b,t}[\delta]$ . The per-frame basis increases the dimensionality of our coefficient output and leads to a slight drop in performance, likely due to a reduced regularizing effect. The common spatial basis, however, suffers a greater performance drop since it also forces kernels in all frames to share the same subspace.

We also compare qualitatively the spatio-temporal kernels produced by our default configuration with those predicted by standard KPN in Figure 2.10. Our model makes better use of the temporal information, applying large weights to pixels across many frames in the burst, whereas KPN tends to overly favor the reference frame. Our network better tracks the

apparent motion in the burst, shifting the kernel accordingly. And it is capable of ignoring outliers caused to excessive motion (all black kernels in Figure 2.10).

*Fixed vs. Burst-specific Basis.* Given that our network predicts both a basis and per-pixel coefficients, a natural question is whether a burst-specific kernel basis is even needed. To address this, we train a network architecture without a basis decoder to only predict coefficients for each burst, and instead learn a basis that is fixed across all bursts in the training set. The fixed basis is learned jointly with this network as a direct learnable tensor. Table 2.4 shows that using a fixed basis in this manner leads to a significant decrease in denoising quality (although still better than standard KPN).

This suggests that while a subspace restriction on kernels is useful, the ideal subspace is scene-dependent and must be predicted adaptively. We further explore this phenomenon in Table 2.5, where we quantify the rank of the predicted bases for individual images, and for pairs of images. Note that the rank can be lower than  $B$ , since we do not effectively require the ‘basis’ vectors  $\{v_b[\cdot, \cdot]\}$  to be linearly independent. We find that the combined rank of basis kernels of image pairs (obtained by concatenating the two bases) is nearly twice the rank obtained from individual images—suggesting limited overlap between the basis sets of different images. We also explicitly compute the average overlap ratio across image pairs as  $1 - \text{rank}(v, v') / [\text{rank}(v) + \text{rank}(v')]$ , and find it to be around 5% on average. This low overlap implies that different bursts do indeed require different bases, justifying our use of burst-specific bases.

**Computational expense.** Next, we evaluate the computational expense of our approach and compare it to the different ablation settings considered in Table 2.4, including standard KPN. We report the total number of floating point operations (FLOPs) required for network prediction and filtering in Table 2.6. We find that in addition to producing higher-quality

	Gain $\propto 1$	Gain $\propto 2$	Gain $\propto 4$	Gain $\propto 8$
$rank(v)$	80.6	81.8	84.3	86.2
$rank(v, v')$	152.2	154.5	159.6	165.2
Overlap ratio	5.4%	5.5%	5.4%	4.4%

Table 2.5: Average basis rank for each noise level (first row), average rank of the union of two bases from random burst pairs (second row), and the average overlap ratio (third row) between the subspaces spanned by the two bases. The low overlap justifies our prediction of a burst-specific basis.

	GFLOPs	Runtime (s)
KPN ( $K = 15$ )	59.3	0.63
Separable ( $K = 15$ )	29.9	0.43
Ours ( $K = 5$ )	28.9	0.24
Ours ( $K = 9$ )	29.1	0.29
Ours ( $B = 10$ )	26.5	0.19
Ours ( $B = 50$ )	28.2	0.27
Ours ( $B = 130$ )	31.7	0.41
<b>Ours (<math>K = 15, B = 90</math>)</b>	<b>29.9</b>	<b>0.30</b>
Common Spatial Basis	40.8	0.49
Per-frame Spatial Basis	41.9	0.57

Table 2.6: FLOPS and runtimes on  $1024 \times 768$  resolution images for different KPN denoising approaches. All variants of our basis prediction network are significantly faster than KPN and match the compute cost of separable filters (with better denoising quality). Increasing the kernel size for our technique comes at marginal cost thanks for the Fourier filtering approach. This allows us to use large kernels for better denoising performance.

results, our approach also requires significantly fewer FLOPs than regular KPN for the same kernel size. This is due to the reduced complexity of our final prediction layer, as well as efficient filtering in the Fourier domain. Also, we find that our approach has nearly identical complexity as separable kernel prediction, while achieving higher denoising performance because it can express a more general class of kernels.

In addition to the evaluation FLOPs, Table 2.6 reports measured running times for the various approaches, benchmarked on a  $1024 \times 768$  image on an NVIDIA 1080Ti GPU. To

Method	Gain $\propto 1$	Gain $\propto 2$	Gain $\propto 4$	Gain $\propto 8$
Direct	38.16	35.39	32.50	30.27
KPN* ( $K = 5$ )	38.86	35.97	32.79	30.01
BPN (ours)	<b>40.16</b>	<b>37.08</b>	<b>33.81</b>	<b>31.19</b>

Table 2.7: Denoising performance on our synthetic color test set. We report performance in terms of Average PSNR (dB). Numbers for KPN\* ( $K = 5$ ) are based on our implementation of [120]. Our method outperforms KPN by more than 1 dB at all noise levels.

compute these timings, we divide the image into  $128 \times 128$  non-overlapping patches to form a batch and send it to the denoising network. Since regular KPN have very high memory requirements, we select the maximum batch size for each method and denoise the entire image in multiple runs. This maximizes GPU throughput. We find that our approach retains its running time advantage over KPN in practice. It is also a little faster than separable kernel prediction—likely due to the improved cache performance we get from using Fourier-domain convolutions with spatially-uniform basis kernels.

**Color burst denoising.** Finally, we report results on a color burst denoising task. We use a similar observation model as (2.5) with noise independently added to each color channel (note this ignores multiplexed measurements and demosaicking). We denoise with separate predicted kernels for each color channel at each location. We extend standard KPN [120] to produce this directly, and modify our method to have the basis decoder produce a “color” kernel basis (of size  $3K^2T \times B$ ), while the coefficient decoder still outputs a  $B$  dimensional coefficient vector.

We use the same training protocol as for grayscale images, using color versions of the Open Images dataset. In this case, training takes 1900k iterations with batches of 8 color images. We construct a new synthetic test set of 100 images from the Open Images validation dataset, with no overlap with our training set. We report comparisons in Table 2.7, showing a similar



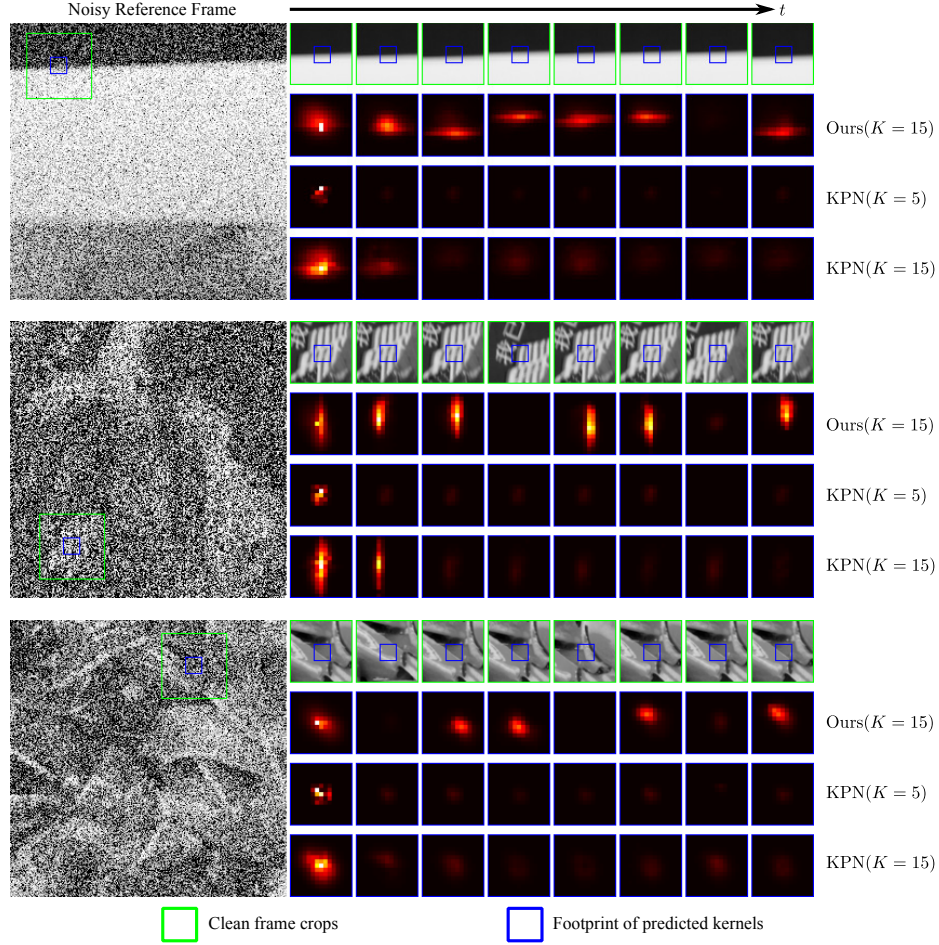


Figure 2.10: We visualize a few 3D kernels predicted by our approach (with  $K = 15$ ), and those produced by standard KPN (with  $K = 5$  and  $K = 15$ ). For kernels predicted at a given location, we also show crops of the different noise-free frames centered at that point, with the support of the kernel marked in blue. In comparison to those from KPN, our kernels are more evenly distributed across all frames in the burst, with spatial patterns that closely follow the apparent motion in the burst.

improvement over KPN [120] as for grayscale images—over 1 dB PSNR at all noise levels. We include qualitative comparisons in Figure 2.11.

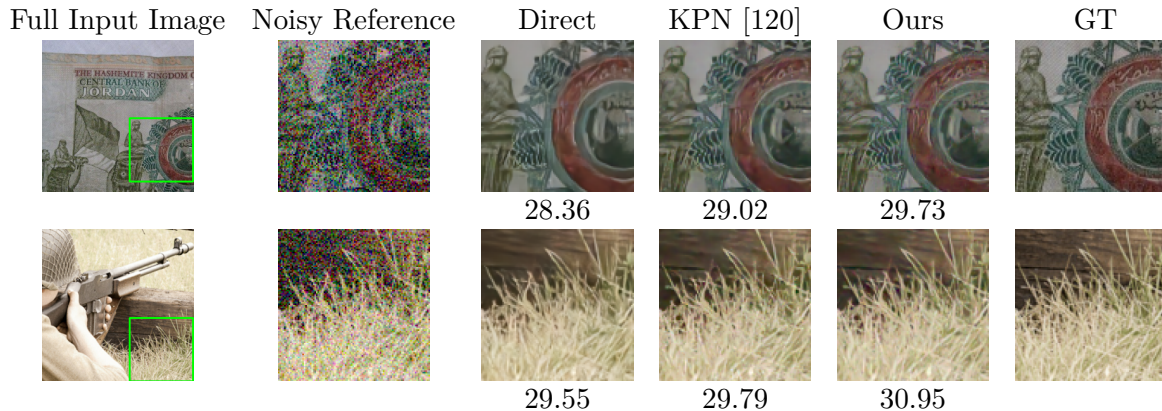


Figure 2.11: We show examples of color denoising using our method on our synthetic color test set, comparing these to direct prediction and our color-extended version of KPN [120] (with  $K = 5$ ). Numbers refer to PSNR (dB) on the full image.

### 2.3.5 Discussion

In this section, we exploit self-similarity to produce a high-quality noise-free image from a sequence of noisy frames captured in low-light environments. We argue that local, per-pixel burst denoising kernels are highly coherent. Based on this, we present a basis prediction network that jointly infers a global, low-dimensional kernel basis and the corresponding per-pixel mixing coefficients that can be used to construct per-pixel denoising kernels. This formulation significantly reduces memory and compute requirements compared to prior kernel-predicting burst denoising methods, allowing us to substantially improve performance by using large kernels, while reducing running time.

While this work focuses on burst denoising, KPN-based methods have been applied to other image and video enhancement tasks including video super-resolution [67], frame interpolation [108, 126, 127], video prediction [45, 66], and video deblurring [193]. All these tasks exhibit similar structure and will likely benefit from our approach.

## 2.4 Exploiting scene appearance in flash photography

### 2.4.1 Introduction

While image denoising algorithms rely on computational models to produce high-quality images through post-processing, flash photography seeks to address the root cause of noise, i.e. insufficient light, by illuminating the scene with a bright burst of light at the time of exposure. It allows the camera to acquire a photograph with a much higher signal-to-noise ratio than would be possible under the dim ambient lighting alone and without introducing any motion or defocus blur. However, flash illumination is not without drawbacks. An on-camera flash often creates unappealing flat shading and harsh shadows, resulting in images that fail to capture the true mood and ambience of the scene.

Researchers have considered combining pairs of flash and no-flash images—captured in quick succession with and without the flash—to create a single enhanced photograph that is both noise-free and accurately represents the scene under ambient lighting. This is achieved by merging information about the ambient scene appearance from the noisy no-flash image, with high-frequency surface image details revealed by the flash [41, 131]. However, these methods assume moderate levels of noise in the no-flash image, and that the flash and no-flash pair are, or can be, aligned.

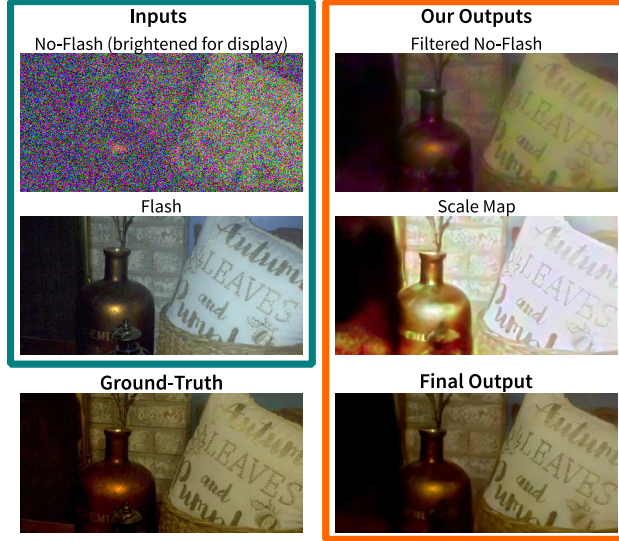


Figure 2.12: Given a pair of images of low-light scenes captured with and without a flash (left), our method produces a high-quality image of the scene under ambient lighting (right). This output is generated by filtering the no-flash image with a predicted field of kernels—to capture a smoothed estimate of scene appearance under ambient lighting, followed by multiplication with a scale map that introduces high-frequency detail illuminated by the flash.

In extremely low light, the no-flash image can be very noisy, especially when using mobile phone cameras with small apertures. This precludes the use of traditional flash/no-flash methods, since the noise obscures even the low-frequency shading information in the no-flash image and makes automatic alignment of the pair unreliable. In comparison, modern neural network-based denoising methods [23, 168, 185, 190] can produce reasonable estimates from a noisy no-flash image alone—although at high noise-levels, they still struggle to reconstruct high-frequency detail.

In this section, we leverage both the ability of modern neural networks to encode strong natural image priors, and the unique combination of appearance information available in a flash and no-flash image pair. Specifically, we consider the task of producing a high-quality image of the scene under ambient lighting given a flash and no-flash pair as input. We focus on extremely low-light scenes such that the no-flash image shows significant noise, and the

appearance of the no-flash image is entirely dominated by the flash illumination. We further assume unknown geometric misalignment between the image pair, due to camera movement typically observed in hand-held burst photography [167].

Under these conditions, we train a deep neural network to take noisy, misaligned no-flash/flash image pairs as input, and output a denoised image of the scene under the scene’s ambient illumination. Rather than directly predicting the denoised image, our network outputs a kernel field used to filter the no-flash image, and a scale map that is multiplied with this filtered output to incorporate high-frequency image details from the flash image. To use the regularizing effect of kernels to effectively filter out the high levels of noise in the no-flash input while overcoming its significant memory and computational costs [115], our network combines the kernel basis prediction approach described in Sec 2.3 with efficient kernel up-sampling. The use of a scale map is inspired by classical flash/no-flash approaches [41, 131] that adopt multiplicative combination based on a view of factorizing images into albedo and shading, where the former is common across the input pair while the latter is not.

We evaluate our approach extensively under different ambient light levels and spatial misalignment, and demonstrate state-of-the-art results for low-light denoising (see example result in Figure 2.12). Our method outperforms denoising without a flash—when using a single or burst of two no-flash images. This demonstrates that a flash input, despite often representing drastically different shading, is still informative towards ambient appearance. Our method also outperforms other standard denoising approaches trained directly on flash/no-flash pairs, highlighting the importance of the formulation and design of our network architecture.

### 2.4.2 Proposed approach

Our goal is to estimate a noise-free color image  $Y[n] \in \mathbb{R}^3$  of the scene under ambient illumination, from a pair of flash and no-flash images  $X_f[n] \in \mathbb{R}^3$  and  $X_{nf}[n] \in \mathbb{R}^3$ , where  $n \in \mathbb{Z}^2$  denotes the pixel location. Since both images are of the same scene, they represent observations of the same surfaces, with the same material properties, but under different illuminations, and with a potential change in viewpoint due to hand motion between the two shots.

**Observation Model and Problem Formulation.** In a chosen reference frame, we denote the appearance of the scene under ambient-only illumination as  $S_a[n] \in \mathbb{R}^3$ , and under flash-only illumination as  $S_f[n] \in \mathbb{R}^3$ . Further, we model the geometric transformations from the reference to the flash and no-flash images as 2D warps  $\mathcal{T}_f(n)$  and  $\mathcal{T}_{nf}(n)$ , respectively. Then, the *noise-free* versions  $\tilde{X}_{nf}[n]$  and  $\tilde{X}_f[n]$  of our no-flash and flash inputs are given by:

$$\begin{aligned}\tilde{X}_{nf}[\mathcal{T}_{nf}(n)] &= S_a[n], \\ \tilde{X}_f[\mathcal{T}_f(n)] &= \alpha_f (S_f[n] + S_a[n]),\end{aligned}\tag{2.10}$$

where  $\alpha_f \leq 1$  is a scalar that captures the effect of a possibly shorter exposure time for the flash image. Note that since the flash is typically much brighter than the ambient lighting ( $S_f[n] \gg S_a[n]$ ), the contribution of the flash-only appearance is dominant in the flash image  $\tilde{X}_f[n]$ .

As in [120] and the previous section, we assume a heteroscedastic Gaussian noise model [46] to account for both read and shot sensor noise. The observed input flash and no-flash pair

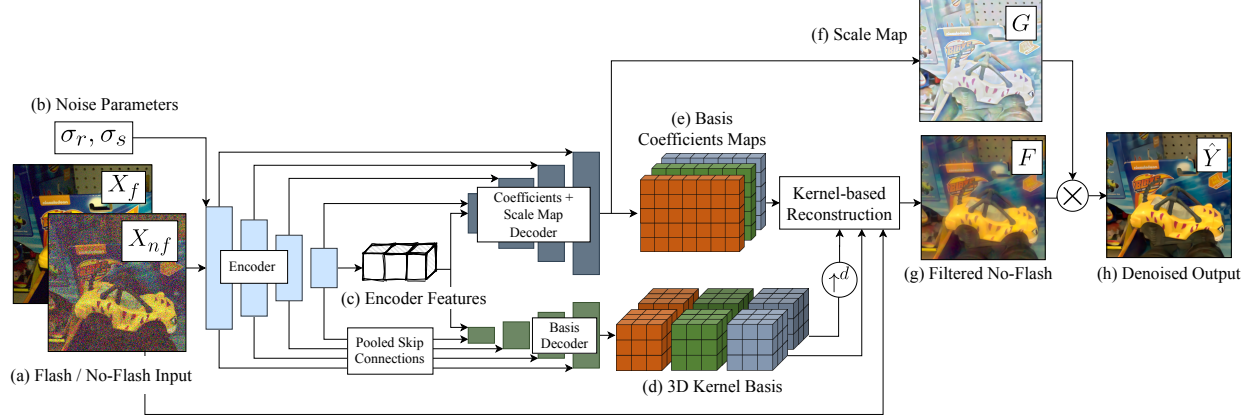


Figure 2.13: **System Overview.** The denoising network takes as input a pair of flash, no-flash images (a) together with the noise parameters (b). After encoding, the resulting features (c) are decoded into a multi-scale basis (d), a set of pixel-wise coefficients (e) and a scale map (f). The no-flash image is filtered using the reconstructed kernels (g) and multiplied by the scale map to produce the final denoised output (h).

relate to their ideal noise-free version (Eq. (2.10)) as:

$$\begin{aligned}
 X_f[n] &\sim \mathcal{N}(\tilde{X}_f[n], \sigma_r^2 + \sigma_s^2 \tilde{X}_f[n]), \\
 X_{nf}[n] &\sim \mathcal{N}(\tilde{X}_{nf}[n], \sigma_r^2 + \sigma_s^2 \tilde{X}_{nf}[n]),
 \end{aligned} \tag{2.11}$$

where  $\sigma_r^2, \sigma_s^2$  are read and shot noise parameters, which we assume are known. Given  $X_f[n]$  and  $X_{nf}[n]$ , and the values of  $\sigma_r^2$  and  $\sigma_s^2$ , we seek to estimate  $Y[n] := S_a[n]$ .

Note that in formulation above, we make a distinction between the target output  $Y[n]$  and the noise-free no-flash image  $\tilde{X}_{nf}[n]$ , because they differ by the warp  $T_{nf}(n)$ . We may wish to use either of the two inputs (flash or no-flash) as the *geometric reference*. If for instance, the no-flash image is the reference, we assume  $T_{nf}(n) = n$  is the identity transformation, and  $Y[n] = \tilde{X}_{nf}[n]$ . Conversely, if we choose the flash image as reference,  $T_f(n) = n$  is chosen to be the identity mapping. In Section 2.4.4, we analyze the effect of this design choice on the output image quality, finding that in most settings, the choice of the no-flash image as reference yields more accurate reconstructions on average.

**Enhancement Network.** We use the basis prediction approach of described in the previous section, which was designed for burst denoising, as the starting point for our model design. Our network differs in two crucial aspects: (a) rather than predicting kernels to filter both the flash and no-flash inputs and summing the result, we filter only the no-flash image and multiply a predicted per-pixel three-channel scale map to form our final output; and (b) we propose an efficient approach to predict larger kernels through upsampling, which is necessary in our setting, because we are filtering a single, highly noisy image. We show an overview of our approach in Figure 2.13.

**Input data.** Our network takes a twelve channel tensor as input, with six channels containing the observed flash  $X_f$  and no-flash  $X_{nf}$  pair (Equation 2.11) themselves, and another six encoding the expected per-pixel standard deviation of noise in these inputs, computed using the (known) values of  $\sigma_s^2$  and  $\sigma_r^2$  and the observed noisy intensities as:  $\sqrt{\sigma_r^2 + \sigma_s^2 \max(0, X^i[n])}$ , for each channel  $i \in \{R, G, B\}$  and  $X = X_f$  and  $X_{nf}$ .

**Predicting a global kernel basis.** Like our burst denoising network, our network features a common encoder whose output is fed to two decoders. The first decoder outputs a global low-rank kernel basis. However, we do not constrain our kernels to be positive and unit-normalized. The second decoder outputs per-pixel mixing coefficients to combine the predicted basis elements and form per-pixel kernels. Another departure from the method in the previous section, the second decode also outputs a 3-channel scale map. We include skip-connections from the encoder to both decoders, using global pooling for connections to the basis decoder.

**Large kernels by interpolation.** A key innovation in our method over our previous burst denoising method is that our basis encodes larger kernels using a 2-scale representation and an interpolation-based reconstruction scheme. This is crucial in our application where these



kernels are used to smooth only one image—the noisy no-flash input—rather than a burst of images as in Section 2.3.

Specifically, our basis decoder outputs a set of  $J$  basis elements, each consisting of a pair of three-channel kernels  $\{(A_j, B_j)\}_{j=1}^J$ , where each  $A_j, B_j \in \mathbb{R}^{K \times K \times 3}$ . We interpret the second kernel  $B_j$  of each pair as a low-frequency term: a large kernel downsampled by a factor  $d$ , with an effective  $((K-1)*d+1) \times ((K-1)*d+1)$  footprint. The  $j^{th}$  element of our basis is then given by  $A_j + (B_j \uparrow^d)$ , where  $\uparrow^d$  denotes bilinear upsampling by a factor  $d$ . So that  $A_j$  can add fine high-frequency details to the kernel center. In our experiments, we use a basis with  $J = 90$  kernels, with a base size  $K = 15$  and upsampling factor of  $d = 4$  resulting in an effective kernel size of  $57 \times 57$ .

**Final reconstruction.** Denoting the per-pixel coefficients from the second decoder as  $\{c_j[n]\}_{j=1}^J$ , we first filter the no-flash input image as:

$$F[n] = \sum_{j=1}^J c_j[n] \left( X_{nf} * (A_j + B_j \uparrow^d) \right) [n]. \quad (2.12)$$

where  $*$  denotes per-channel convolution between three-channel images and kernels. Note that the filtering with upsampled kernels can be carried out efficiently, by pre-filtering the no-flash image and using dilated convolutions:

$$F[n] = \sum_{j=1}^J c_j[n] \left( (X_{nf} * A_j)[n] + (X_{nf}^h *_d B_j)[n] \right), \quad (2.13)$$

where  $X_{nf}^h[n] = (X_{nf} * h)[n]$  is the result of smoothing the no-flash input with a  $(2d-1) \times (2d-1)$  tent kernel  $h[n]$ , and  $*_d$  represents dilated convolution with a factor of  $d$ .

The result  $F[n]$  of this filtering step will typically encode a noise-free (and in the case of the flash as reference, an aligned) estimate of scene appearance under ambient illumination. However, due to the lower signal-to-noise ratio of  $X_{nf}$ , this filtering step cannot recover the high-frequency details that are illuminated only resolved in the flash image. To recover these, our full-pixel decoder also produces a scale map  $G[n] \in \mathbb{R}^3$ . Our final output  $\hat{Y}[n]$  is given by the element-wise product of this scale map and the filtered no-flash image:

$$\hat{Y}[n] = F[n] \odot G[n]. \quad (2.14)$$

This formulation is inspired by classic flash/no-flashing denoising methods [41, 131] that add high-frequency details from the flash image in the *log domain*, i.e., corresponding to a product in our linear domain. In Section 2.4.4, we show this outperforms the alternative of using kernels to jointly denoise the no-flash and flash images.

**Training details.** While our network accepts raw linear sensor measurements as input and produces an estimate of linear intensities in  $Y[n]$ , it is trained to maximize image quality in a color and gamma-corrected sRGB space. In particular, we assume that for each training sample  $(X_f^t, X_{nf}^t, Y^t)$ , we also have a scalar gain  $\alpha^t$  (representing a desired target brightness level), and a  $3 \times 3$  color transform matrix  $C^t$  based on camera sensor parameters and white-balance settings, such that the mapping to sRGB is given by  $f_t(Y[n]) = \gamma(\alpha C^t \hat{Y}[n])$ , where  $\gamma(\cdot)$  is a gamma correction curve.

We train our model to minimize the sum of a squared  $L_2$  pixel loss, and a  $L_1$  gradient loss between the estimated and ideal rendered images:

$$L = \frac{1}{T} \sum_{t=1}^T \|f_t(\hat{Y}_t) - f_t(Y_t)\|^2 + \eta |\partial_x * (f_t(\hat{Y}_t) - f_t(Y_t))| + \eta |\partial_y * (f_t(\hat{Y}_t) - f_t(Y_t))|, \quad (2.15)$$

where  $\partial_x$  and  $\partial_y$  are horizontal and vertical gradient filters.

We train our model using the Adam optimizer [72], beginning with a learning rate of  $10^{-4}$ , and going through two learning rate drops every time validation loss saturates, for a total of roughly 1.5 million iterations.

### 2.4.3 Experimental setup

We now describe our experimental setup to evaluate our approach.

**Dataset.** We use the dataset of Aksoy *et al.* [2], which contains 16-bit well-exposed ambient-only and flash-only image pairs. These images were crowdsourced from users who were asked to capture images with hand-held mobile phones in real-world settings, and roughly had a 0.5-1 second delay between captures of the pair. We split the dataset as follows: 2519 images for the training set, 128 for validation and 128 for testing, considering  $440 \times 440$  crops (random crops for training, and fixed central crops for validation and testing). We simulate a real low-light capture by dimming the linear ambient-only image by dividing with a random factor in  $[2, 50]$ , sampled uniformly in the log domain. This forms our *no-flash* input. We increase the exposure of the flash-only image by a constant factor 2, and add it to the no-flash input to obtain our *flash* input.

**Misalignment and simulated noise.** The image pairs provided by [2] were automatically aligned by finding correspondences with feature matching. Since this would be unrealistic in low-light images, we undo such alignment by warping the no-flash or flash image (the other is the reference) with a random homography. To obtain the homography parameters, we assume the camera’s FOV is 90 degrees to get its intrinsic matrix. We perturb it with a random 3D rotation uniformly sampled in the range  $[-0.5, 0.5]$  degrees in each axis, followed by random 2D scaling by a factor uniformly sampled in  $[0.98, 1.02]$ , and a random 2D translation of  $[0, 2]$  pixels. The overall average per-pixel displacement between our flash and no-flash inputs ranges up to 20 pixels (Manhattan distance). Note that real-world non-idealities like parallax, occlusions, blur, etc. originally present in the data are preserved by undoing [2]’s alignment.

We use the same noise parameters for the flash and no-flash image, i.e., we assume they were captured with the same ISO setting. During training, we randomly sample the noise parameters  $\sigma_r$  and  $\sigma_s$  uniformly in the log-domain in the ranges:  $\log(\sigma_r) \in [-3, -2]$  and  $\log(\sigma_s) \in [-4, -2.6]$ .

**Losses and metrics.** The preprocessing pipeline is executed on the original linear color space of the camera. To compute losses, we set the desired gain  $\alpha^t$  in Section 2.4.2 to be the inverse of the factor we used to dim the image above, since the original images in [2] were well-lit. The database also includes a color transform matrix for each image which we use as  $C^t$ . We evaluate performance by computing PSNR and SSIM between the rendered versions of our estimate and the ground-truth.

**Baselines.** We compare to denoising without a flash input: using a single no-flash image denoised by a version of our architecture (without a scale map), and a burst of two (misaligned) no-flash inputs denoised using the state-of-the-art burst denoising method described in

Method	100x Dimmed	50x Dimmed	25x Dimmed	12.5x Dimmed
No-flash input only				
Our Architecture	24.91	27.23	29.31	30.98
2-frame burst input (no flash)				
BPN [169]	25.58	27.75	29.65	31.21
Flash and no-flash input pair				
Direct Prediction	24.80	27.06	29.12	30.84
KPN [120]	25.87	27.94	29.69	31.21
BPN [169]	26.11	28.04	29.75	31.21
<b>Ours</b>	<b>26.75</b>	<b>28.56</b>	<b>30.14</b>	<b>31.52</b>

Table 2.8: **Quantitative results.** Thanks to the richer signal provided by the flash input, our method outperforms our single image denoising baseline, and a 2-frame burst denoising baseline. Comparisons to standard burst denoising approaches adapted to use flash–no-flash pairs show that our model architecture with its filtering/scale decomposition and larger kernels outperforms previous work. These results hold over a wide range of ambient light levels, shown here as dimming factors between the low-light no-flash input and a well-lit ground-truth target.

Section 2.3 (which we refer to as BPN). For flash and no-flash image inputs, we compare our method to other standard architectures: a direct prediction network which simply regresses to the denoised output, and burst denoising methods KPN [120] and BPN applied to the flash and no-flash pair. All of these methods were trained on our dataset, and provided information about noise standard deviation in an identical manner to our method.

#### 2.4.4 Evaluation

We compare our approach to baseline methods for both denoising without a flash input, and to applying existing network architectures to a flash and no-flash pair. We also include ablations describing the effect of our kernel interpolation approach, and of choosing the no-flash vs. flash image as geometric reference.

We begin by evaluating our method, choosing the ambient image as geometric reference (i.e., we assume  $T_{nf}(n) = n$ ), on our test set of 128 images, and comparing it to the various

baselines described above. We fix the noise level to  $\log(\sigma_r) = -2.6$  and  $\log(\sigma_s) = -3.6$ , sample a random homography for each pair to be applied to the flash image (for the no-flash burst, this homography is applied to the second no-flash image), and repeat our evaluation with a discrete set of dimming factors:  $[100, 50, 25, 12.5]$ . Note that the factor 100 lies outside our training range, and demonstrates the robustness of our method.

Our method consistently outperforms all methods, regardless of the dimming factor, as seen by the quantitative results in Table 2.8. We also include example reconstructions in Figure 2.15, where we see that our method reconstructs fine surface detail with higher fidelity than the other methods.

In Figure 2.14, we take a closer look at the effect of misalignment. We take a subset of 64 flash and no-flash pairs from our test set (all dimmed with a factor of 50), and evaluate each set with different homographies that cause different average pixel displacements. We plot the PSNR of reconstruction by various methods for different degrees of displacement (for the single no-flash input baseline, these numbers are the same for all displacements). As expected, the accuracy of all methods decreases with greater misalignment. Nevertheless, we find that our method consistently outperforms all baselines, including the single no-flash input even with misalignment greater than 10 pixels.

**Ablation.** In our method section, we considered two options for the alignment reference: with the output geometrically aligned with the flash input, or the no-flash image. In Table 2.8, we reported results with the no-flash input as the reference (for our method, as well as the other methods evaluated on the flash and no-flash pair). This was based on an evaluation of both alternatives, which we report in Table. 2.9.

Setting	Flash Reference	No-Flash Reference	No-Flash w/o $\{B_j\}$
100x dimmed	<b>26.83 dB</b>	26.75 dB	26.45 dB
50x dimmed	28.39 dB	<b>28.56 dB</b>	28.42 dB
25x dimmed	29.55 dB	<b>30.14 dB</b>	30.09 dB
12.5x dimmed	30.45 dB	<b>31.52 dB</b>	31.51 dB

Table 2.9: **Ablation study.** We compare the performance of our method to two ablations. One uses the flash image instead of the no-flash image as reference for the geometric transformation. The other uses a kernel basis without interpolation, leading to an effective kernel size of only  $15 \times 15$ .

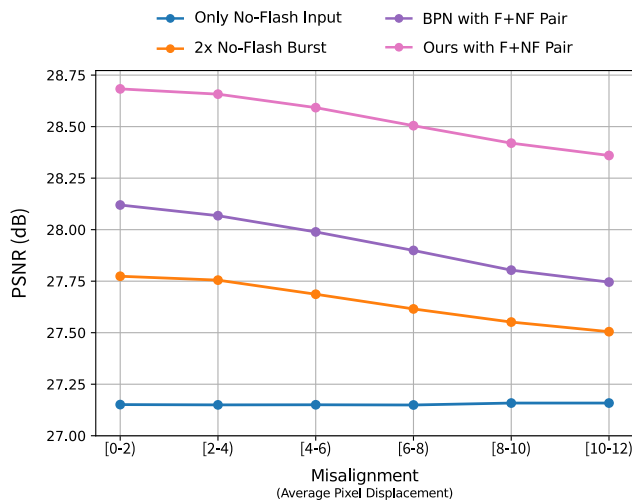


Figure 2.14: **Performance vs. misalignment.** We show the performance profile of our method and select baselines as a function of average displacement between the two frames. Our model consistently delivers superior performance and is robust to large misalignment between its inputs.

We found that except for the lowest light level, the using the no-flash image as reference yields results that are quantitatively better (this is also true for the other baselines). However, looking at the actual reconstructions in Figure 2.17, we find both images to be of similar visual quality—with the lower quantitative performance of the flash reference being largely due to slight, and largely imperceptible, alignment errors in low-frequency shading. However, we do find that using the flash image as reference sometimes yields visually sharper results.

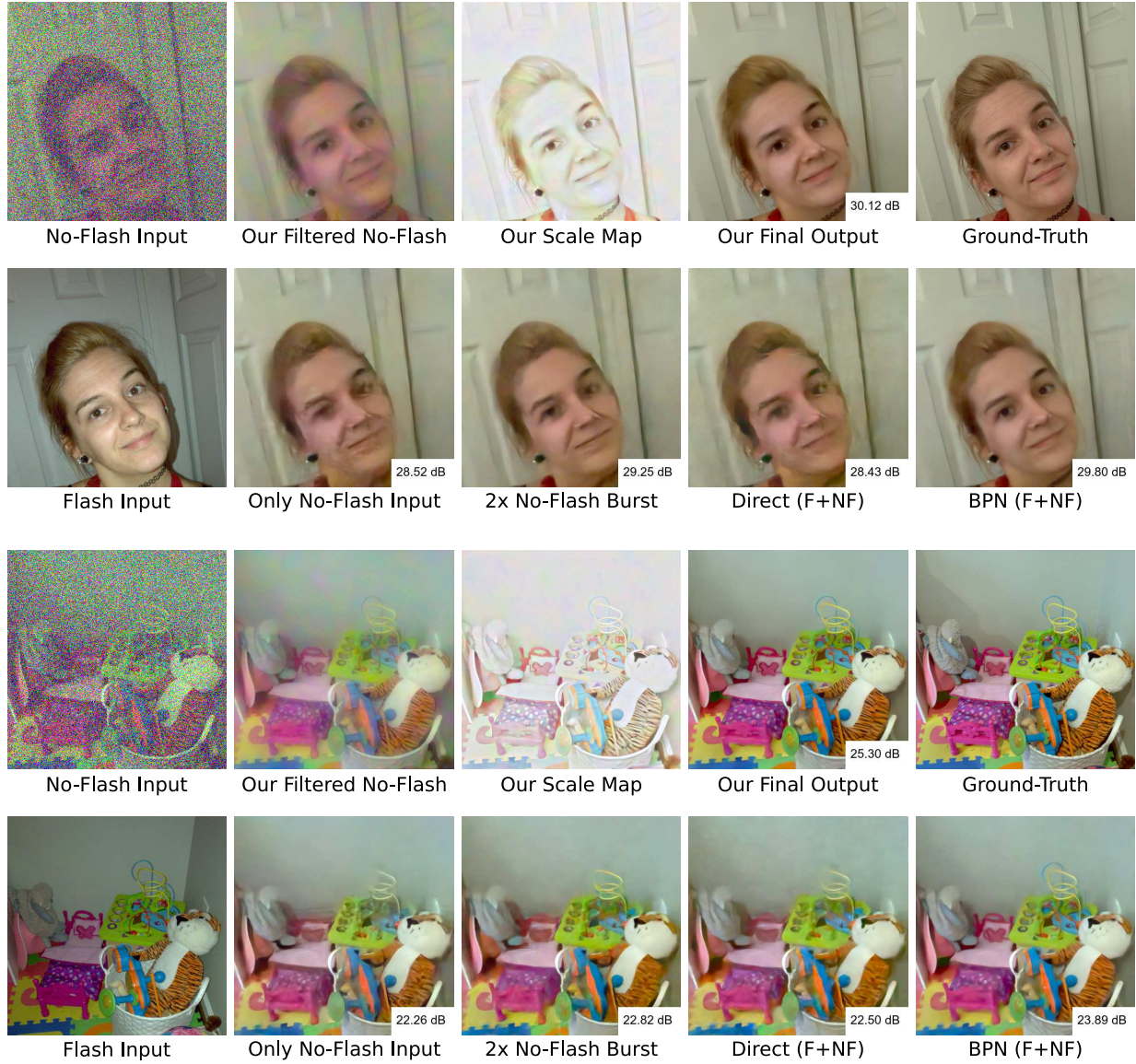


Figure 2.15: **Qualitative comparison.** Our method uses flash/no-flash image pairs to denoise low-light images. It produces cleaner outputs than baseline flash/no-flash denoisers (*Direct (F+NF)*, *BPN (F+NF)*), as well as single-image (*Only No-Flash Input*) and burst denoisers (*2x No-Flash Burst*). We also visualize our intermediate filtered no-flash image and scale map.



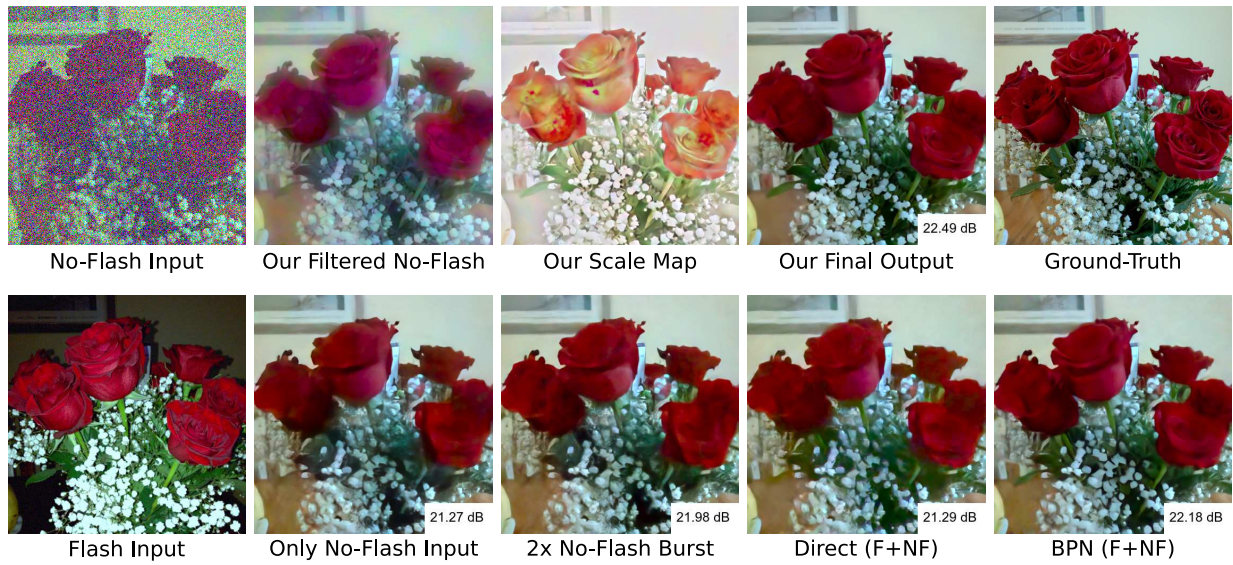


Figure 2.16: **Qualitative comparison (continued)**. Our method uses flash/no-flash image pairs to denoise low-light images. It produces cleaner outputs than baseline flash/no-flash denoisers (*Direct (F+NF)*, *BPN (F+NF)*), as well as single-image (*Only No-Flash Input*) and burst denoisers (*2× No-Flash Burst*). We also visualize our intermediate filtered no-flash image and scale map.

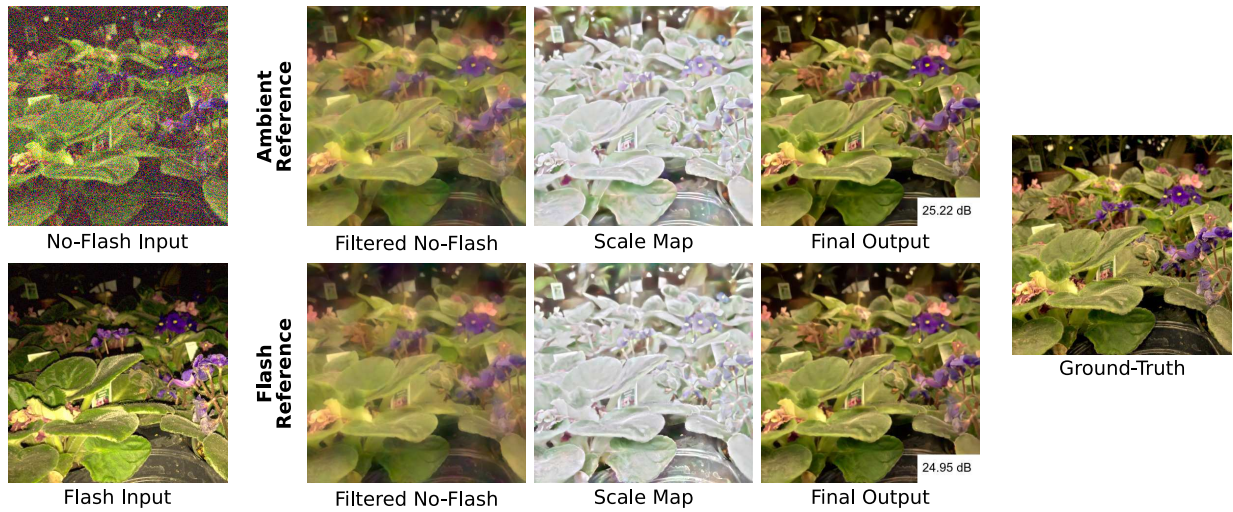


Figure 2.17: **Flash vs. no-flash as reference frame**. We use the ambient-only image as the reference frame for our reconstruction (*top*), i.e. the ground truth is aligned to the no-flash image. We found this choice leads to a lower error on average, compared to the alternative, using the flash as reference (*bottom*).

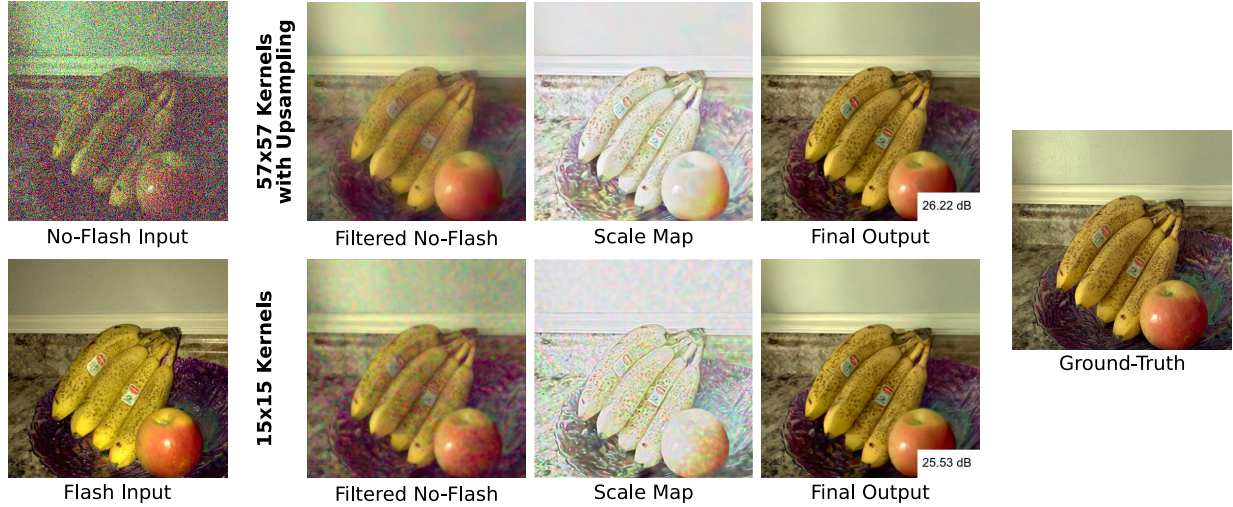


Figure 2.18: **Benefit of large kernels.** By using a 2-scale kernel decomposition, where the low-pass component is bilinearly upsampled, our model (*top*) can better denoise the ambient-only image. This leads to reduced residual chroma noise, which makes the scale map more effective at recovering fine details. Without it (*bottom*), the kernels are too small to effectively denoise the ambient image, so the scale map needs to compensate for the residual mid-frequency noise.

Table 2.9 also evaluates the benefit of using larger filters through our interpolation-based approach. We find that by allowing filters with a larger footprint ( $57 \times 57$ ), our two-scale kernel basis improves denoising quality, especially at low light levels. As shown in Figure 2.18, large kernels yield a smoother filtering of the noisy no-flash image, so that the flash-driven scale map does not need to overcompensate for residual mid-frequency color noise, leading to better reconstructions in the final output.

### 2.4.5 Discussion

This section introduced a method to effectively leverage the unique mix of visual information available in a flash and no-flash image pair, and produce high-quality images in low-light environments. Our method preserves the warmth and colors of the ambient lighting while bringing out fine details thanks to the flash image. Drawing on traditional flash/no-flash

techniques, our network architecture assembles its output from a filtered ambient-only image, and a scale map that encoded high-frequency details from the flash. Although it was not trained with any intermediate supervision, we found our network automatically learns to carry out both the necessary geometric alignment between the frames, and the photometric transfer needed to produce state-of-the-art reconstructions.

# Chapter 3

## Learning without direct supervision for computational photography

Many computational photography tasks are about decoding scene properties (including color, illumination, geometry and material properties), from observations that are photographs of the scene taken with real cameras . On the most basic level, computational photography seek to build a “computational super camera” to output a photograph of the scene that has enhanced quality from a single image (or a very few images). Due to limited capabilities of a real camera, these input images could be degraded observations that are noisy, low-resolution, blurry or has low dynamic range etc., and require a restoration algorithm to recover corresponding un-corrupted image. On the next level of computational photography, the goal is to go beyond the capability of this super camera and reverse the process of how an image is formed to decompose a photograph into light fields, different illumination components, geometric information of the scene and material properties. Example applications include inferring lighting, depth, surface normal or reflectance from one or several photographs of a scene.

At both levels, deep convolutional neural networks (CNNs) have recently emerged as an effective tool for such tasks of decoding scene properties from photographs [20, 28, 35, 43, 49, 79, 95, 145, 146, 183, 185]. Specifically, a CNN for a given application is trained on a large dataset that consists of pairs of ground-truth scene maps and observed images (in many cases where the degradation or the image formation process is well characterized, such dataset can be synthesized or rendered from ground-truth images or scene properties). This training set allows the CNN to learn to exploit the expected statistical properties of scene maps in that application domain, to solve what is essentially an ill-posed inverse problem.

However, unlike semantic tasks where ground-truth estimations can be easily annotated by human beings, for many domains of computational photography, it is impractical or prohibitively expensive to capture ground-truth scene properties, and construct such a large representative training set. Unfortunately, it is often in such domains that a computational photography solution is most useful. For example, collecting a large-scale real image restoration dataset is very challenging because for real noisy, (motion-) blurry images it is often too expensive to capture their corresponding high-quality images, and therefore leads to unsatisfying performance for methods that are trained on synthetic dataset when deployed in practice.

On the other hand, it is often practical and straightforward for us to capture indirect measurements of those scene properties that are of interest. These indirect measurements are different samples of the scene via photographs. Such captured images may have a different exposure, focus, view, illumination, or instant of capture. Each image is a projection of the scene which contains partial information about the scene property. Examples include blurry images of the same scene taken at different instants (therefore have different camera or scene motion), or images taken under different illumination and from different views that have the same underlying geometry and reflectance. In this chapter, we study how to use such indirect

measurements to train deep neural networks to decode scene properties without using any ground-truth scene maps. We demonstrate that these measurements could provide indirect but sufficient supervision to learn the statistics of scene properties. This provides a valuable and practical alternative to the training of neural networks for computational photography.

Specifically, we discuss how to train deep neural networks with indirect supervision for tasks at these two different aforementioned levels of computational photography. In Section 3.1, we describe how to train image enhancement models with only low-dimensional or degraded measurements. In Section 3.2, we propose a data capturing hardware system and a training strategy to train a normal estimation model to output high-quality normal maps with only images and low-quality stereo.

## **3.1 Training image estimators without ground-truth images**

### **3.1.1 Introduction**

Reconstructing images from imperfect observations is a classic inference task in many computational photography applications. In compressive sensing [36], a sensor makes partial measurements for efficient acquisition. These measurements correspond to a low-dimensional projection of the higher-dimensional image signal, and the system relies on computational inference for recovering the full-dimensional image. In other cases, cameras capture degraded images which are then recovered by restoration algorithms [48, 181, 198]. Among these algorithms, deep CNNs have recently become the state-of-the-art in almost every image reconstruction application by training on a large dataset. However, collecting a large-scale dataset of consisting of pairs of ground-truth images and observed measurements is often



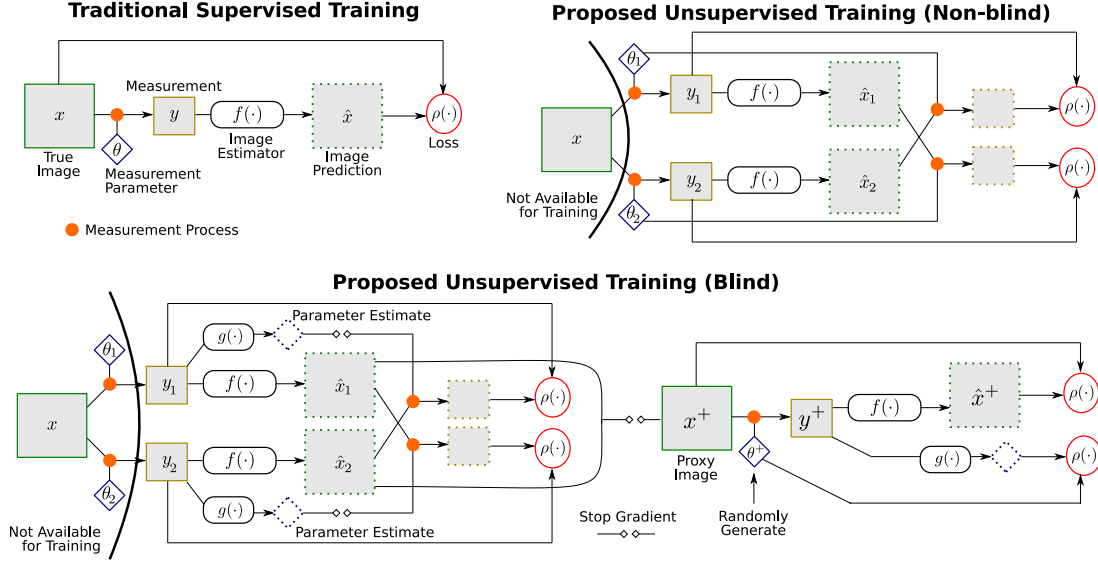


Figure 3.1: **Unsupervised Training from Measurements.** Our method allows training image estimation networks  $f(\cdot)$  from sets of pairs of varied measurements, but without the underlying ground-truth images. (*Top Right*) We supervise training by requiring that network predictions from one measurement be consistent with the other, when measured with the corresponding parameter. (*Bottom*) In the blind training setting, when both the image and measurement parameters are unavailable, we also train a parameter estimator  $g(\cdot)$ . Here, we generate a proxy training set from the predictions of the model (as it is training), and use synthetic measurements from these proxies to supervise training of the parameter estimator  $g(\cdot)$ , and augment training of the image estimator  $f(\cdot)$ .

impractical. Recently, Lehtinen *et al.* [87] proposed a solution to this issue for denoising, with a method that trains with only pairs of noisy observations. While their method yields remarkably accurate network models without needing any ground-truth images for training, it is applicable only to the specific case of estimation from noisy measurements—when each image intensity is observed as a sample from a (potentially unknown) distribution with mean or mode equal to its corresponding true value.

In this section, we introduce an unsupervised method for training image estimation networks that can be applied to a general class of observation models—where measurements are a linear function of the true image, potentially with additive noise. As training data, it only

requires two observations for the same image but not the underlying image itself<sup>2</sup>. The two measurements in each pair are made with different parameters (such as different compressive measurement matrices or different blur kernels), and these parameters vary across different pairs. Collecting such a training set provides a practical alternative to the more laborious one of collecting full image ground-truth. Given these measurements, our method trains an image estimation network by requiring that its prediction from one measurement of a pair be consistent with the other measurement, when observed with the corresponding parameter. With sufficient diversity in measurement parameters for different training pairs, we show this is sufficient to train an accurate network model despite lacking direct ground-truth supervision.

While our method requires knowledge of the measurement *model* (e.g., blur by convolution), it also incorporates a novel mechanism to handle the blind setting during training—when the measurement *parameters* (e.g., the blur kernels) for training observations are unknown. To be able to enforce consistency as above, we use an estimator for measurement parameters that is trained simultaneously using a “proxy” training set. This set is created on-the-fly by taking predictions from the image network even as it trains, and pairing them with observations synthetically created using randomly sampled, and thus known, parameters. The proxy set provides supervision for training the parameter estimator, and to augment training of the image estimator as well. This mechanism allows our method to nearly match the accuracy of fully supervised training on image and parameter ground-truth.

We validate our method with experiments on image reconstruction from compressive measurements and on blind deblurring of face images, with blind and non-blind training for the latter, and compare to fully-supervised baselines with state-of-the-art performance. The supervised baselines use a training set of ground-truth images and generate observations with

---

<sup>2</sup>Note that at test time, the trained network only requires one observation as input as usual.



random parameters on the fly in each epoch, to create a much larger number of effective image-measurement pairs. In contrast, our method is trained with only two measurements per image from the same training set (but not the image itself), with the pairs kept fixed through all epochs of training. Despite this, our unsupervised training method yields models with test accuracy close to that of the supervised baselines, and thus presents a practical way to train CNNs for image estimation when lacking access to image ground truth.

### 3.1.2 Related work

**CNN-based Image Estimation.** Many imaging tasks require inverting the measurement process to obtain a clean image from the partial or degraded observations—denoising [17], deblurring [181], super-resolution [48], compressive sensing [36], etc. While traditionally solved using statistical image priors [44, 137, 198], CNN-based estimators have been successfully employed for many of these tasks. Most methods [20, 28, 35, 79, 122, 146, 183, 185] learn a network to map measurements to corresponding images from a large training set of pairs of measurements and ideal ground-truth images. Some learn CNN-based image priors, as denoisers [22, 134, 185] or GANs [3], that are agnostic to the inference task (denoising, deblurring, etc.), but still tailored to a chosen class of images. All these methods require access to a large domain-specific dataset of ground-truth images for training. However, capturing image ground-truth is burdensome or simply infeasible in many settings (e.g., for MRI scans [110] and other biomedical imaging applications). In such settings, our method provides a practical alternative by allowing estimation networks to be trained from measurement data alone.

**Unsupervised Learning.** Unsupervised learning for CNNs is broadly useful in many applications where large-scale training data is hard to collect. Accordingly, researchers have proposed unsupervised and weakly-supervised methods for such applications, such as depth

estimation [52, 194], intrinsic image decomposition [94, 113], etc. However, these methods are closely tied to their specific applications. In this work, we seek to enable unsupervised learning for image estimation networks. In the context of image modeling, Bora *et al.* [14] propose a method to learn a GAN model from only degraded observations. Their method, like ours, includes a measurement model with its discriminator for training (but requires knowledge of measurement parameters, while we are able to handle the blind setting). Their method proves successful in training a generator for ideal images. We seek a similar unsupervised means for training image reconstruction and restoration networks.

The closest work to ours is the recent *Noise2Noise* method of Lehtinen *et al.* [87], who propose an unsupervised framework for training denoising networks by training on pairs of noisy observations of the same image. In their case, supervision comes from requiring the denoised output from one observation be close to the other. This works surprisingly well, but is based on the assumption that the expected or median value of the noisy observations is the image itself. We focus on a more general class of observation models, which requires injecting the measurement process in loss computation. We also introduce a proxy training approach to handle blind image estimation applications.

Also related are the works of Metzler *et al.* [119] and Zhussip *et al.* [196], that use Stein’s unbiased risk estimator for unsupervised training from only measurement data, for applications in compressive sensing. However, these methods are specific to estimators based on D-AMP estimation [118], since they essentially train denoiser networks for use in unrolled AMP iterations for recovery from compressive measurements. In contrast, ours is a more general framework that can be used to train generic neural network estimators.

### 3.1.3 Proposed approach

Given a measurement  $y \in \mathbb{R}^M$  of an ideal image  $x \in \mathbb{R}^N$  that are related as

$$y = \theta x + \epsilon, \quad (3.1)$$

our goal is to train a CNN to produce an estimate  $\hat{x}$  of the image from  $y$ . Here,  $\epsilon \sim p_\epsilon$  is random noise with distribution  $p_\epsilon(\cdot)$  that is assumed to be zero-mean and independent of the image  $x$ , and the parameter  $\theta$  is an  $M \times N$  matrix that models the linear measurement operation. Often, the measurement matrix  $\theta$  is structured with fewer than  $MN$  degrees of freedom based on the measurement model—e.g., it is block-Toeplitz for deblurring with entries defined by the blur kernel. We consider both non-blind estimation when the measurement parameter  $\theta$  is known for a given measurement during inference, and the blind setting where  $\theta$  is unavailable but we know the distribution  $p_\theta(\cdot)$ . For blind estimators, we address both non-blind and blind training—when  $\theta$  is known for each measurement in the training set but not at test time, and when it is unknown during training as well.

Since (3.1) is typically non-invertible, image estimation requires reasoning with the statistical distribution  $p_x(\cdot)$  of images for the application domain, and conventionally, this is provided by a large training set of typical ground-truth images  $x$ . In particular, CNN-based image estimation methods train a network  $f : y \rightarrow \hat{x}$  on a large training set  $\{(x_t, y_t)\}_{t=1}^T$  of pairs of corresponding images and measurements, based on a loss that measures error  $\rho(\hat{x}_t - x_t)$  between predicted and true images across the training set. In the non-blind setting, the measurement parameter  $\theta$  is known and provided as input to the network  $f$  (we omit this in the notation for convenience), while in the blind setting, the network must also reason about the unknown measurement parameter  $\theta$ .

To avoid the need for a large number of ground-truth training images, we propose an unsupervised learning method that is able to train an image estimation network using measurements alone. Specifically, we assume we are given a training set of two measurements  $(y_{t:1}, y_{t:2})$  for each image  $x_t$ :

$$y_{t:1} = \theta_{t:1} x_t + \epsilon_{t:1}, \quad y_{t:2} = \theta_{t:2} x_t + \epsilon_{t:2}, \quad (3.2)$$

but not the images  $\{x_t\}$  themselves. We require the corresponding measurement parameters  $\theta_{t:1}$  and  $\theta_{t:2}$  to be different for each pair, and further, to also vary across different training pairs. These parameters are assumed to be known for the non-blind training setting, but not for blind training.

**Unsupervised Training for Non-Blind Image Estimation.** We begin with the simpler case of non-blind estimation, when the parameter  $\theta$  for a given measurement  $y$  is known, both during inference and training. Given pairs of measurements with known parameters, our method trains the network  $f(\cdot)$  using a “swap-measurement” loss on each pair, defined as:

$$\mathcal{L}_{\text{swap}} = \frac{1}{T} \sum_t \rho(\theta_{t:2} f(y_{t:1}) - y_{t:2}) + \rho(\theta_{t:1} f(y_{t:2}) - y_{t:1}). \quad (3.3)$$

This loss evaluates the accuracy of the full images predicted by the network from each measurement in a pair, by comparing it to the other measurement—using an error function  $\rho(\cdot)$ —after simulating observation with the corresponding measurement parameter. Note *Noise2Noise* [87] can be seen as a special case of (3.3) for measurements are degraded only by noise, with  $\theta_{t:1} = \theta_{t:2} = I$ .

When the parameters  $\theta_{t:1}, \theta_{t:2}$  used to acquire the training set are sufficiently diverse and statistically independent for each underlying  $x_t$ , this loss provides sufficient supervision to

train the network  $f(\cdot)$ . To see this, we consider using the  $L_2$  distance for the error function  $\rho(z) = \|z\|^2$ , and note that (3.3) represents an empirical approximation of the expected loss over image, parameter, and noise distributions. Assuming the training measurement pairs are obtained using (3.2) with  $x_t \sim p_x$ ,  $\theta_{t:1}, \theta_{t:2} \sim p_\theta$ , and  $\epsilon_{t:1}, \epsilon_{t:2} \sim p_\epsilon$  drawn i.i.d. from their respective distributions, we have

$$\begin{aligned} \mathcal{L}_{\text{swap}} &\approx 2 \mathbb{E}_{x \sim p_x} \mathbb{E}_{\theta_1 \sim p_\theta} \mathbb{E}_{\epsilon_1 \sim p_\epsilon} \mathbb{E}_{\theta_2 \sim p_\theta} \mathbb{E}_{\epsilon_2 \sim p_\epsilon} \|\theta_2 f(\theta_1 x + \epsilon_1) - (\theta_2 x + \epsilon_2)\|^2 \\ &= 2\sigma_\epsilon^2 + 2 \mathbb{E}_{x \sim p_x} \mathbb{E}_{\theta \sim p_\theta} \mathbb{E}_{\epsilon \sim p_\epsilon} \left( f(\theta x + \epsilon) - x \right)^T Q \left( f(\theta x + \epsilon) - x \right), \quad Q = \mathbb{E}_{\theta' \sim p_\theta} (\theta'^T \theta'). \end{aligned} \quad (3.4)$$

Therefore, because the measurement matrices are independent, we find that in expectation the swap-measurement loss is equivalent to supervised training against the true image  $x$ , with an  $L_2$  loss that is weighted by the  $N \times N$  matrix  $Q$  (upto an additive constant given by noise variance). When the matrix  $Q$  is full-rank, the swap-measurement loss will provide supervision along all image dimensions, and will reach its theoretical minimum ( $2\sigma_\epsilon^2$ ) *iff* the network makes exact predictions.

The requirement that  $Q$  be full-rank implies that the distribution  $p_\theta$  of measurement parameters must be sufficiently diverse, such that the full *set* of parameters  $\{\theta\}$ , used for training measurements, together span the entire domain  $\mathbb{R}^N$  of full images. Therefore, even though measurements made by individual  $\theta$ —and even pairs of  $(\theta_{t:1}, \theta_{t:2})$ —are incomplete, our method relies on the fact that the full set of measurement parameters used during training *is* complete. Indeed, for  $Q$  to be full-rank, it is important that there be no systematic deficiency in  $p_\theta$  (e.g., no vector direction in  $\mathbb{R}^N$  left unobserved by all measurement parameters used in training). Also note that while we derived (3.4) for the L2 loss, the argument applies to any error function  $\rho(\cdot)$  that is minimized only when its input is 0.

In addition to the swap loss, we also find it useful to train with an additional “self-measurement” loss that measures consistency between an image prediction and its own corresponding input measurement:

$$\mathcal{L}_{\text{self}} = \frac{1}{T} \sum_t \rho\left(\theta_{t:1} f(y_{t:1}) - y_{t:1}\right) + \rho\left(\theta_{t:2} f(y_{t:2}) - y_{t:2}\right). \quad (3.5)$$

While not sufficient by itself, we find the additional supervision it provides to be practically useful in yielding more accurate network models since it provides more direct supervision for each training sample. Therefore, our overall unsupervised training objective is a weighted version of the two losses  $\mathcal{L}_{\text{swap}} + \gamma \mathcal{L}_{\text{self}}$ , with weight  $\gamma$  chosen on a validation set.

**Unsupervised Training for Blind Image Estimation.** We next consider the more challenging case of blind estimation, when the measurement parameter  $\theta$  for an observation  $y$  is unknown—and specifically, the blind training setting, when it is unknown even during training. The blind training setting complicates the use of our unsupervised losses in (3.3) and (3.5), since the values of  $\theta_{t:1}$  and  $\theta_{t:2}$  used there are unknown. Also, blind estimation tasks often have a more diverse set of possible parameters  $\theta$ . While supervised training methods with access to ground-truth images can generate a very large database of synthetic image-measurement pairs by pairing the same image with many different  $\theta$  (assuming  $p_\theta(\cdot)$  is known), our unsupervised framework has access only to two measurements per image.

However, in many blind estimation applications (such as deblurring), the parameter  $\theta$  has comparatively limited degrees of freedom and the distribution  $p_\theta(\cdot)$  is known. Consequently, it is feasible to train estimators for  $\theta$  from an observation  $y$  with sufficient supervision. With these assumptions, we propose a “proxy training” approach for unsupervised training of blind image estimators. This approach treats estimates from our network during training as a source of image ground-truth to train an estimator  $g : y \rightarrow \hat{\theta}$  for measurement parameters.

We use the image network’s predictions to construct synthetic observations as:

$$x_{t:i}^+ \leftarrow f(y_{t:i}), \quad \theta_{t:i}^+ \sim p_\theta, \quad \epsilon_{t:i}^+ \sim p_\epsilon, \quad y_{t:i}^+ = \theta_{t:i}^+ x_{t:i}^+ + \epsilon_{t:i}^+, \quad \text{for } i \in \{1, 2\}, \quad (3.6)$$

where  $\theta_{t:i}^+$  and  $\epsilon_{t:i}^+$  are sampled on the fly from the parameter and noise distributions, and  $\leftarrow$  indicates an assignment with a “stop-gradient” operation (to prevent loss gradients on the proxy images from affecting the image estimator  $f(\cdot)$ ). We use these synthetic observations  $y_{t:i}^+$ , with known sampled parameters  $\theta_{t:i}^+$ , to train the parameter estimation network  $g(\cdot)$  based on the loss:

$$\mathcal{L}_{\text{prox}:\theta} = \frac{1}{T} \sum_t \sum_{i=1}^2 \rho(g(y_{t:i}^+) - \theta_{t:i}^+). \quad (3.7)$$

As the parameter network  $g(\cdot)$  trains with augmented data, we simultaneously use it to compute estimates of parameters for the original observations:  $\hat{\theta}_{t:i} \leftarrow g(y_{t:i})$ , for  $i \in \{1, 2\}$ , and compute the swap- and self-measurement losses in (3.3) and (3.5) on the original observations using these estimated, instead of true, parameters. Notice that we use a stop-gradient here as well, since we do not wish to train the parameter estimator  $g(\cdot)$  based on the swap- or self-measurement losses—the behavior observed in (3.4) no longer holds in this case, and we empirically observe that removing the stop-gradient leads to instability and often causes training to fail.

In addition to training the parameter estimator  $g(\cdot)$ , the proxy training data in (3.6) can be used to augment training for the image estimator  $f(\cdot)$ , now with *full supervision* from the proxy images as:

$$\mathcal{L}_{\text{prox}:x} = \frac{1}{T} \sum_t \sum_{i=1}^2 \rho(f(y_{t:i}^+) - x_{t:i}^+). \quad (3.8)$$

This loss can be used even in the non-blind training setting, and provides a means of generating additional training data with more pairings of image and measurement parameters. Also note that although our proxy images  $x_{t:i}^+$  are approximate estimates of the true images, they

represent the ground-truth for the synthetically generated observations  $y_{t:i}^+$ . Hence, the losses  $\mathcal{L}_{\text{prox}:\theta}$  and  $\mathcal{L}_{\text{prox}:x}$  are approximate only in the sense that they are based on images that are not sampled from the true image distribution  $p_x(\cdot)$ . And the effect of this approximation diminishes as training progresses, and the image estimation network produces better image predictions (especially on the training set).

Our overall method randomly initializes the weights of the image and parameter networks  $f(\cdot)$  and  $g(\cdot)$ , and then trains them with a weighted combination of all losses:  $\mathcal{L}_{\text{swap}} + \gamma \mathcal{L}_{\text{self}} + \alpha \mathcal{L}_{\text{prox}:\theta} + \beta \mathcal{L}_{\text{prox}:x}$ , where the scalar weights  $\alpha, \beta, \gamma$  are hyper-parameters determined on a validation set. For non-blind training (of blind estimators), only the image estimator  $f(\cdot)$  needs to be trained, and  $\alpha$  can be set to 0.

### 3.1.4 Experiments

We evaluate our framework on two well-established tasks: non-blind image reconstruction from compressive measurements, and blind deblurring of face images. These tasks were chosen since large training sets of ground-truth images *is* available in both cases, which allows us to demonstrate the effectiveness of our approach through comparisons to fully supervised baselines.

**Reconstruction from Compressive Measurements.** We consider the task of training a CNN to reconstruct images from compressive measurements. We follow the measurement model of [79, 183], where all non-overlapping  $33 \times 33$  patches in an image are measured individually by the same low-dimensional orthonormal matrix. Like [79, 183], we train CNN models that operate on individual patches at a time, and assume ideal observations without noise. We train models for compression ratios of 1%, 4%, and 10% (using corresponding matrices provided by [79]).



Table 3.1: Performance (in PSNR dB) of various methods for compressive measurement reconstruction, on BSD68 and Set11 images for different compression ratios.

Method	Supervised	BSD68			Set11		
		1%	4%	10%	1%	4%	10%
TVAL3 [89]	<b>✗</b>	-	-	-	16.43	18.75	22.99
BM3D-AMP [118] (patch-wise)	<b>✗</b>	-	-	-	5.21	18.40	22.64
BM3D-AMP [118] (full-image)	<b>✗</b>	-	-	-	5.59	17.18	23.07
ReconNet [79]	<b>✓</b>	-	21.66	24.15	17.27	20.63	24.28
ISTA-Net+ [183]	<b>✓</b>	19.14	22.17	25.33	17.34	21.31	<u>26.64</u>
Supervised Baseline (Ours)	<b>✓</b>	<b>19.74</b>	<b>22.94</b>	<b>25.57</b>	<b>17.88</b>	<b>22.61</b>	<b>26.74</b>
Unsupervised Training (Ours)	<b>✗</b>	<u>19.67</u>	<u>22.78</u>	<u>25.40</u>	<u>17.84</u>	<u>22.20</u>	26.33
Unsupervised Training (Ours) <i>ablation without self-loss</i>	<b>✗</b>	19.59	22.73	25.32	17.80	22.10	26.16

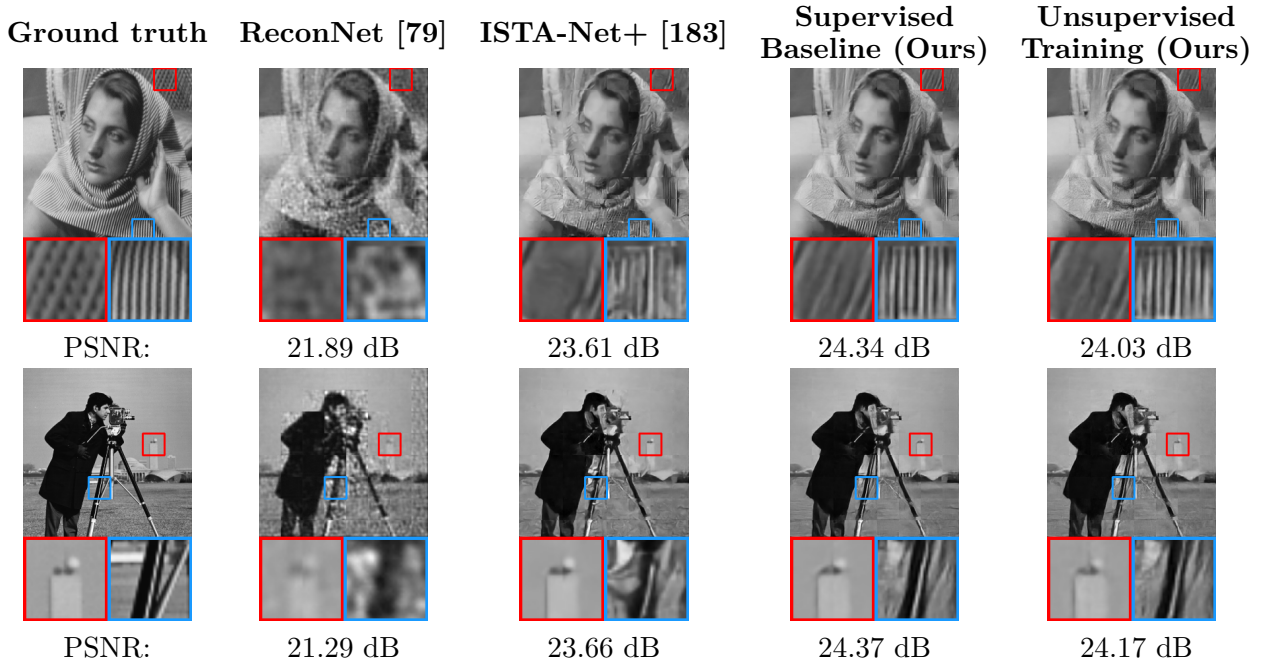


Figure 3.2: Images reconstructed by various methods from compressive measurements (at 10% ratio).

We generate a training and validation set, of 100k and 256 images respectively, by taking  $363 \times 363$  crops from images in the ImageNet database [140]. We use a CNN architecture that stacks two U-Nets [136], with a residual connection between the two. We begin by training our architecture with full supervision, using *all overlapping* patches from the training images,

and an  $L_2$  loss between the network’s predictions and the ground-truth image patches. For unsupervised training with our approach, we create two partitions of the original image, each containing *non-overlapping* patches. The partitions themselves overlap, with patches in one partition being shifted from those in the other. We measure patches in both partitions with the same measurement matrix, to yield two sets of measurements. These provide the diversity required by our method as each pixel is measured with a different patch in the two partitions. Moreover, this measurement scheme can be simply implemented in practice by camera translation. The shifts for each image are randomly selected, but kept fixed throughout training. Since the network operates independently on patches, it can be used on measurements from both partitions. To compute the swap-measurement loss, we take the network’s individual patch predictions from one partition, arrange them to form the image, and extract and then apply the measurement matrix to shifted patches corresponding to the other partition. The weight  $\gamma$  for the self-measurement loss is set to 0.05 based on the validation set.

In Table 3.1, we report results for existing compressive sensing methods that use supervised training [79, 183], as well as two methods that do not require any training [89, 118]. We report numbers for these methods from the evaluation in [183] that, like us, reconstruct each patch in an image individually. We also report results for the algorithm in [118] by running it on entire images (i.e., using the entire image for regularization while still using the per-patch measurement model). Note that [118] is a D-AMP-based estimator (and while slower, performs similarly to the learned D-AMP estimators proposed in [119, 196] as per their own evaluation).

Evaluating our fully supervised baseline against these methods, we find that it achieves state-of-the-art performance. We then report results for training with our unsupervised framework, and find that this leads to accurate models that only lag our supervised baseline

by 0.4 db or less in terms of average PSNR on both test sets—and in most cases, actually outperforms previous methods. This is despite the fact that these models have been trained without any access to ground-truth images. In addition to our full unsupervised method with both the self- and swap- losses, Table 3.1 also contains an ablation without using the self-loss, which is found to lead to a slight drop in performance. Figure 3.2 provides example reconstructions for some images, and we find that results from our unsupervised method are extremely close in visual quality to those of the baseline model trained with full supervision.

**Blind Face Image Deblurring.** We next consider the problem of blind motion deblurring of face images. Like [146], we consider the problem of restoring  $128 \times 128$  aligned and cropped face images that have been affected by motion blur, through convolution with motion blur kernels of size upto  $27 \times 27$ , and Gaussian noise with standard deviation of two gray levels. We use all 160k images in the CelebA training set [107] and 1.8k images from Helen training set [83] to construct our training set, and 2k images from CelebA val and 200 from the Helen training set for our validation set. We use a set of 18k and 2k random motion kernels for training and validation respectively, generated using the method described in [20]. We evaluate our method on the official blurred test images provided by [146] (derived from the CelebA and Helen test sets). Note that unlike [146], we do not use any semantic labels for training.

In this case, we use a single U-Net architecture to map blurry observations to sharp images. We again train a model for this architecture with full supervision, generating blurry-sharp training pairs on the fly by pairing random of blur kernels from training set with the sharp images. Then, for unsupervised training with our approach, we choose two kernels for each training image to form a training set of measurement pairs, that are kept fixed (including the added Gaussian noise) across all epochs of training. We first consider non-blind training, using

the true blur kernels to compute the swap- and self-measurement losses. Here, we consider training with and without the proxy loss  $\mathcal{L}_{\text{prox}:x}$  for the network. Then, we consider the blind training case where we also learn an estimator for blur kernels, and use its predictions to compute the measurement losses. Instead of training a entirely separate network, we share the initial layers with the image UNet, and form a separate decoder path going from the bottleneck to the blur kernel. The weights  $\alpha, \beta, \gamma$  are all set to one in this case.

We report results for all versions of our method in Table 3.2, and compare it to [146], as well as a traditional deblurring method that is not trained on face images [173]. We find that with full supervision, our architecture achieves state-of-the-art performance. Then with non-blind training, we find that our method is able to come close to supervised performance when using the proxy loss, but does worse without—highlighting its utility even in the non-blind setting. Finally, we note that models derived using blind-training with our approach are also able to produce results nearly as accurate as those trained with full supervision—despite lacking access both to ground truth image data, and knowledge of the blur kernels in their training measurements. Figure 3.3 illustrates this performance qualitatively, with example deblurred results from various models on the official test images. We also visualize the blur kernel estimator learned during blind training with our approach in Figure 3.4 on images from our validation set.

### 3.1.5 Discussion

In this section, we presented an unsupervised method to train image estimation networks from only measurements pairs, without access to ground-truth images, and in blind settings, without knowledge of measurement parameters. We validated this approach on well-established tasks where sufficient ground-truth data (for natural and face images) was available, since it allowed us to compare to training with full-supervision and study the performance gap between the

Table 3.2: Performance of various methods on blind face deblurring on test images from [146].

Method	Supervised	Helen		CelebA	
		PSNR	SSIM	PSNR	SSIM
Xu <i>et al.</i> [173]	✗	20.11	0.711	18.93	0.685
Shen <i>et al.</i> [146]	✓	25.99	0.871	25.05	0.879
Supervised Baseline (Ours)	✓	<b>26.13</b>	<b>0.886</b>	<b>25.20</b>	<b>0.892</b>
Unsupervised Non-blind (Ours)	✗	25.95	0.878	25.09	0.885
Unsupervised Non-blind (Ours) <i>without proxy loss</i>	✗	25.47	0.867	24.64	0.873
Unsupervised Blind (Ours)	✗	25.93	0.876	25.06	0.883

supervised and unsupervised settings. But we believe that our method’s real utility will be in opening up the use of CNNs for image estimation to new domains—such as medical imaging, applications in astronomy, etc.—where such use has been so far infeasible due to the difficulty of collecting large ground-truth datasets.





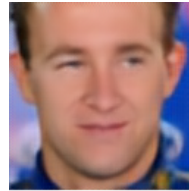
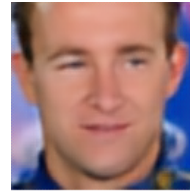

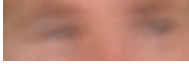
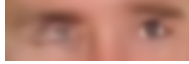
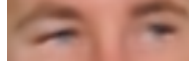
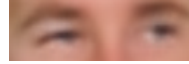
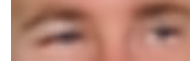







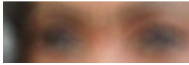





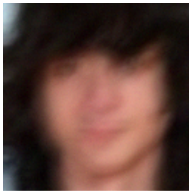
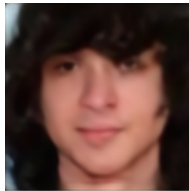
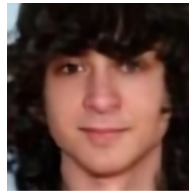
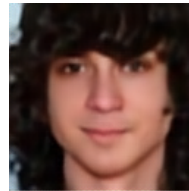
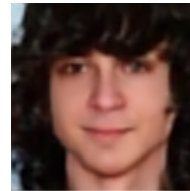

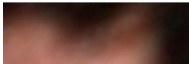
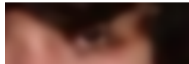
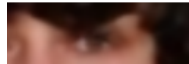
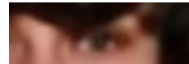
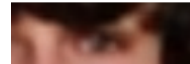

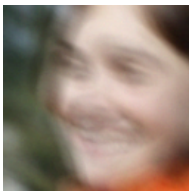
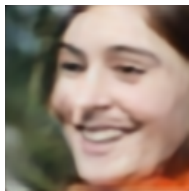

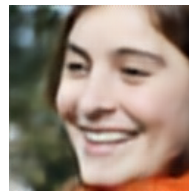
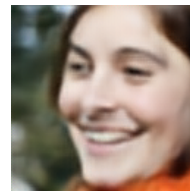
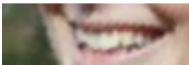
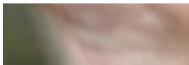
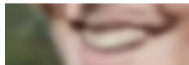


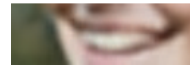
GT	Input	Shen <i>et al.</i> [146]	Supervised (Ours)	Non-blind (Ours)	Blind (Ours)
					
					
PSNR:		22.69 dB	24.61 dB	25.16 dB	25.19 dB
					
					
PSNR:		26.83 dB	28.18 dB	28.27 dB	28.16 dB
					
					
PSNR:		26.59 dB	28.29 dB	27.42 dB	26.77 dB
					
					
PSNR:		22.36 dB	23.50 dB	22.84 dB	22.94 dB

Figure 3.3: Blind face deblurring results using various methods. Results from our unsupervised approach, with both non-blind and blind training, nearly match the quality of the supervised baseline.

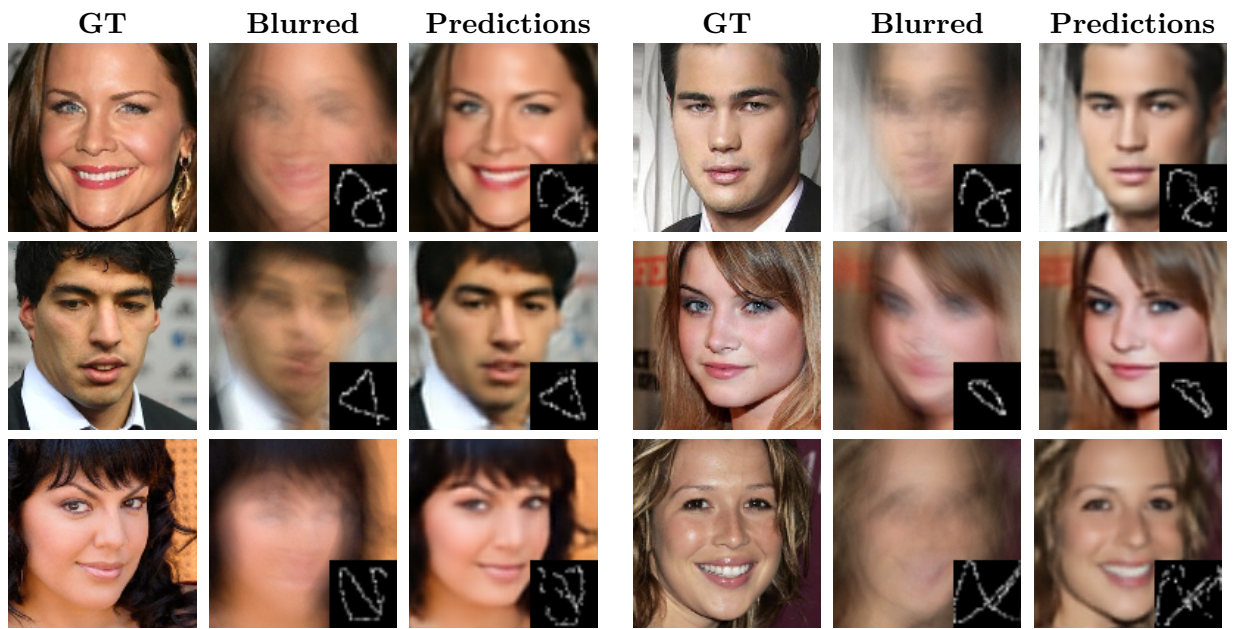


Figure 3.4: Image and kernel predictions on validation images. We show outputs of our model’s kernel estimator, that is learned as part of blind training to compute swap- and self-measurement losses.

## 3.2 Training a dark flash normal camera without ground-truth normals

### 3.2.1 Introduction

In casual mobile photography, images are often captured under poor lighting conditions. This is especially problematic for applications that try to reconstruct geometry and reflectance of the scene from these captured images. Controlling the visible lighting or supplementing it with a flash is often too difficult or too disruptive to be practical. On the other hand, the near infrared (NIR) lighting in a scene can be much more easily controlled and is invisible to the user. In this section, we seek to use a single “dark flash” NIR image and a single visible image taken under uncontrolled lighting to recover high quality maps of the surface normals, diffuse albedos, and specular intensities in the scene. Specifically, we focus on faces - the most common photography subject at the short ranges over which active illumination is effective.

We present a deep neural network that takes as input one RGB image captured under uncontrolled visible lighting and one monochrome NIR image captured from the same viewpoint, but under controlled lighting provided by a single source located near the camera. The network generates a surface normal and reflectance estimate (diffuse albedo + specular intensity) at each pixel.

However, collecting a large-scale dataset of high-quality ground-truth normals can be very expensive. Therefore, we propose to train this network by combining two imperfect but complementary cues: a stereo depth map that provides a reliable estimate of the low-frequency components of the scene’s 3d shape, along with RGB and NIR images under different illumination and from different views containing photometric cues that convey



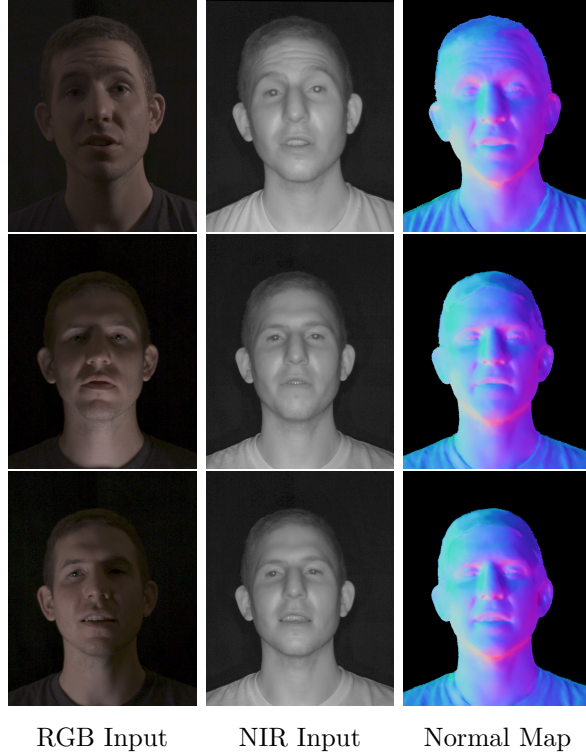


Figure 3.5: Estimating surface geometry from a single RGB image is challenging. We augment this input with a single NIR “dark flash” image captured at the same time, and present a network that can estimate high quality normal maps and reflectance maps (not shown) under a wide range of visible lighting conditions.

higher-frequency geometric details. These measurements are far easier to obtain than ground truth geometry and appearance measurements. We also propose a hardware setup for collecting this data.

We compare our technique to a baseline learning approach that uses only a single RGB image as input and state-of-the-art methods for single image intrinsic image decomposition [145] and relighting [124]. We are able to produce overall more stable and more accurate outputs even in very challenging visible light conditions. We also present two applications of integrating our technique in a mobile photography pipeline: optimizing depths computed by an independent stereo technique and reducing shadows in an image post-capture.

### 3.2.2 Related work

**Intrinsic imaging and shape from shading.** Decomposing a single image into its underlying shape and reflectance is a classical under-constrained problem in computer vision [7, 62]. One class of methods employ hand-designed priors, learned from relatively small datasets [5, 6] or NIR imagery [30], to disambiguate these components. Learning-based methods have been proposed more recently that train convolutional neural networks to perform this task using rendered datasets [96, 147], sparse human annotations [10], or multi-view images under different lighting conditions [180]. Whereas some learning approaches function as “black boxes” [147], others incorporate a physically-based image formation model [8, 96, 145, 154]. Similiar to ours, other approaches explore network inputs beyond a single RGB image, including an additional visible flash image and a depth map [132] or a single NIR image [179].

A number of methods are specifically designed to work on images of faces. This includes 3D morphable models [12], which are commonly used as a prior on reflectance and geometry in learning-based approaches [145, 150, 155]. Sanyal et al. [141] estimate the shape of a face within a single image in the form of blending weights over a parametric face model. Similar to our approach, other techniques estimate dense normal or displacement maps [182] including for faces partially hidden by occluders [34, 157]. However these methods do not attempt to disentangle reflectance data from shading.

In contrast to these prior techniques, we propose a neural network that takes a single front-lit NIR image in addition to a color input image, enabling our technique to perform well even in very challenging visible light conditions. Our use of controlled NIR lighting provides a number of benefits. First, the ambient NIR light in a scene is often weak or completely absent in indoor environments and is significantly attenuated by atmospheric absorption

outdoors, making it practical to control this aspect of a scene. Second, it results in a more tractable estimation problem in contrast to single-image “shape from shading” and intrinsic image decomposition techniques that must simultaneously reason about shape, material properties, and lighting. Third, it provides a stable source of information about the shape and appearance of the scene even under very challenging visible lighting. By locating the NIR light source near the camera, this setup minimizes shadows in the scene while produces specular highlights along surfaces that are nearly perpendicular to the viewing direction, giving a useful cue for determining surface orientations. Our training process is also novel in the way that it combines two independent and complimentary signals.

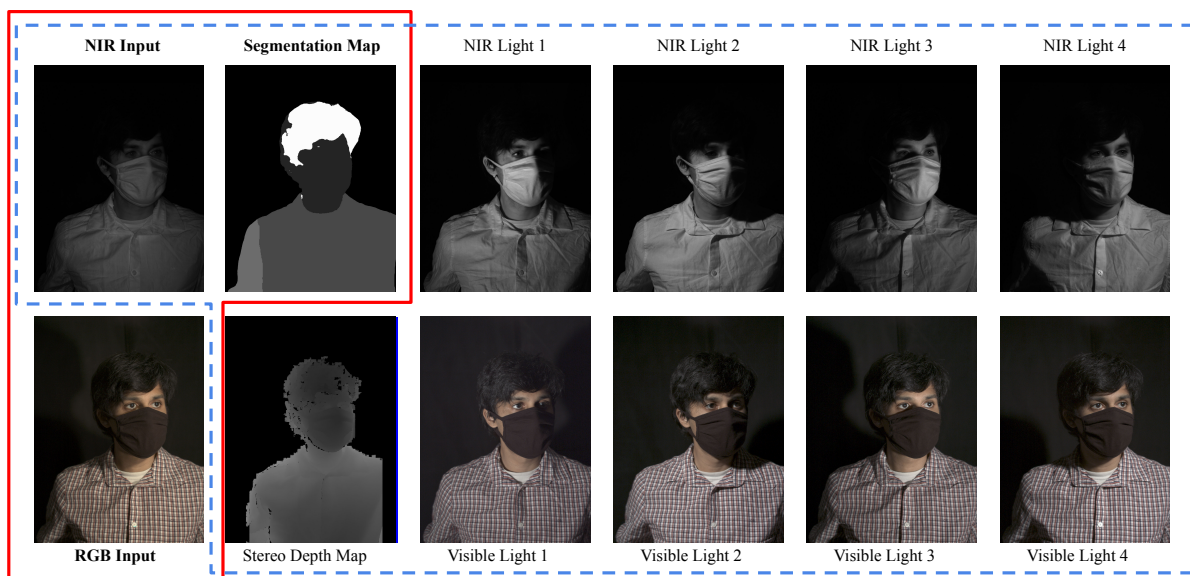


Figure 3.6: Our network learns to estimate shape and reflectance from a single front-lit NIR image, a single RGB image under arbitrary lighting, and a semantic segmentation map computed from the RGB image (inputs are enclosed by the red line). During training we also use a stereo depth map and replace the RGB image under arbitrary lighting with 4 RGB+NIR image pairs captured under calibrated point lights (the training inputs are inside the blue dashed line).

**Fusing depth and normals.** Depth estimated from methods like stereo triangulation and normals estimated from shading cues are complementary measurements for shape recovery. Nehab et al. [123] describe a technique that seeks to combine the more accurate low-frequency information provided by direct depth measurement techniques with the higher-frequency geometric details provided by photometric measurements. We use their technique to evaluate how our approach could be used to improve a stereo pipeline (Section 3.2.5). More recent work poses this as an optimization problem that seeks a surface that best agrees with these different signals [4, 19, 31, 97, 177]. While our method does not use any depth information at inference time, our training method is similar to these approaches in that we also combine a stereo and photometric loss term.

**Face relighting.** Most single image face relighting methods include some representation of shape and reflectance as intermediate components. Our network architecture is similar to the one proposed by Nestmeyer et al. [124] for simulating lighting changes in a single image assumed to have been captured under a single directional light. Zhou et al. [192] present a dataset of relit portrait images generated using single-image normal and illumination estimates and a Lambertian reflectance model. Although surface geometry is fundamental to relighting, it is also possible to train an end-to-end network that does not explicitly reason about shape [153]. We similarly use multiple images of a scene captured under varying controlled lighting to train our network in order to enable a much simpler set of inputs for inference.

**Combining infrared and color imagery.** A NIR (and/or ultraviolet) dark flash image can be used to denoise a color image captured in low visible light conditions [77], or serve as

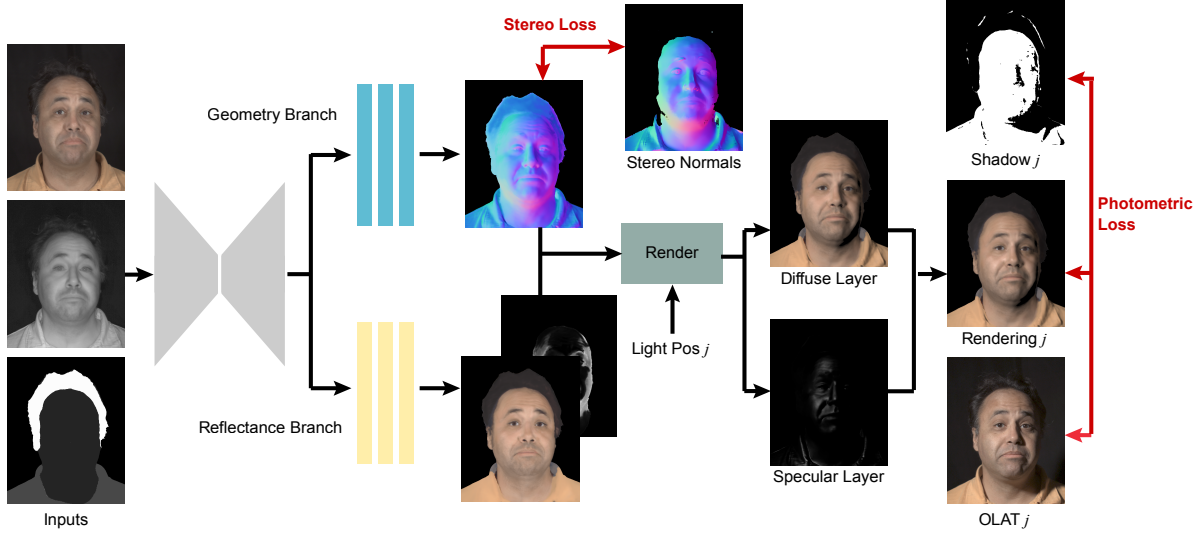


Figure 3.7: Illustration of our network and training strategy. We estimate network weights that minimize a photometric loss, computed between images rendered from our network outputs and ground truth images captured under known lighting, and a stereo loss, driven by differences between the output normals and those estimated using an independent stereo technique.

a guide for correcting motion blur [174]. Techniques have also been developed that employ controlled NIR lighting to simulate better visible lighting in real-time video communication systems [55, 160]. We see these as compelling potential applications of this work.

### 3.2.3 Network design and training

Our goal is to estimate a normal map and a reflectance map from a single RGB image and a front-lit “dark flash” NIR image. We train a deep neural network to perform this task. As an auxiliary input, we use a 6-class semantic segmentation map computed from the RGB image (background, head, hair, body, upper arm and lower arm) [24]. We found this segmentation map was a useful cue for helping the network reason about shape and reflectance. An example set of inputs are shown in Figure 3.6 (red line).

Our training procedure is driven in part by a physically-based image formation model that connects the outputs of our network to images of a scene taken under known point lighting. This image formation model combines a standard Lambertian diffuse term with the Blinn-Phong BRDF [13], which has been used to model the specular reflectance of human skin [166]. Specifically, we introduce a reflectance function  $f$  that gives the ratio of reflected light to incident light for a particular unit-length light vector  $\mathbf{l}$ , view vector  $\mathbf{v}$ , surface normal  $\mathbf{n}$ , four-channel (RGB+NIR) albedo  $\boldsymbol{\alpha}$ , scalar specular intensity  $\rho$ , and specular exponent  $m$ :

$$f(\mathbf{l}, \mathbf{v}, \mathbf{n}) = \boldsymbol{\alpha} + \rho \frac{m+2}{2\pi} (\mathbf{n} \cdot \mathbf{h})^m, \quad (3.9)$$

where  $\mathbf{h} = (\mathbf{n} + \mathbf{l}) / \|\mathbf{n} + \mathbf{l}\|$ . The observed intensity at a pixel due to a point light is given by

$$I(\cdot) = f(\mathbf{l}, \mathbf{v}, \mathbf{n}) (\mathbf{n} \cdot \mathbf{l}) L, \quad (3.10)$$

the product of the reflectance, cosine term, and light intensity  $L$ . We do not observe the reflected intensity from enough unique light directions at each pixel to estimate all of the parameters in Equation 3.9. We therefore fix the specular exponent to  $m = 30$  based on prior measurements of human skin [166] and our own observations, and estimate only  $\mathbf{n}$ ,  $\boldsymbol{\alpha}$ , and  $\rho$ . The geometric quantities  $\mathbf{l}$  and  $\mathbf{v}$ , and light intensity  $L$  are determined by the calibration procedures described in Section 3.2.4.

Illustrated in Figure 3.7, we use a standard UNet with skip connections [135]. The encoder and decoder each consist of 5 blocks with 3 convolutional layers per block. The bottleneck has 256 channels. The output of this UNet is forwarded to two separate networks: a geometry branch that predicts a normal map  $\tilde{\mathbf{n}}$ , and a reflectance branch that predicts an albedo map  $\tilde{\boldsymbol{\alpha}}$  and log-scale specular intensity map,  $\log(\tilde{\rho})$ . Both branches have 3 convolutional layers with 32 channels and one final output layer.

We do not rely on ground truth normals or reflectance data to supervise training. Instead we combine a stereo loss and a photometric loss derived from data that is far easier to obtain: four one-light-at-a-time (OLAT) images in both RGB and NIR of the same subject, in the same exact pose, illuminated by a set of calibrated lights activated individually in rapid succession, and a stereo depth map (blue dashed line in Figure 3.6). These images are only used at training time.

A stereo loss encourages our estimated normals  $\tilde{\mathbf{n}}$  to agree with the gradients of the stereo depth map  $\mathbf{n}_s$ . The gradients are computed by applying a 5x5 Prewitt operator on stereo depth maps that are smoothed with RGB-guided bilateral filtering. Similar to [179], our stereo loss combines a L1 vector loss and angular loss:

$$\mathcal{L}_s(\tilde{\mathbf{n}}) = \|\tilde{\mathbf{n}} - \mathbf{n}_s\|_1 - (\tilde{\mathbf{n}} \cdot \mathbf{n}_s). \quad (3.11)$$

A photometric loss is computed between each of the OLAT images and an image rendered according to Equation 3.10 and our network outputs for the corresponding lighting condition:

$$\mathcal{L}_p^j(\tilde{\mathbf{n}}, \tilde{\boldsymbol{\alpha}}, \tilde{\rho}) = \left\| S_j \odot \left( I(\mathbf{l}_j, \mathbf{v}, \tilde{\mathbf{n}}, \tilde{\boldsymbol{\alpha}}, \tilde{\rho}) - \hat{I}_j \right) \right\|_1, \quad (3.12)$$

where  $\hat{I}_j$  is the per-pixel color observed in the  $j^{\text{th}}$  OLAT image, and  $S_j$  is a binary shadow map, computed by raycasting using the stereo depth and calibrated light position (Section 3.2.4).

We also apply a prior to the albedo map that encourages piecewise constant variation [6]:

$$\mathcal{L}_c(\tilde{\boldsymbol{\alpha}}) = \sum_i \sum_{j \in \mathcal{N}(i)} \|\tilde{\boldsymbol{\alpha}}_i - \tilde{\boldsymbol{\alpha}}_j\|_1, \quad (3.13)$$

where  $\mathcal{N}(i)$  is the  $5 \times 5$  neighborhood centered at pixel  $i$ . We apply this prior only to clothing pixels, those labeled as either body or arms in the segmentation mask. We found that other regions in the scene did not benefit from this regularization.

Our total loss function is a weighted sum of these terms:

$$\mathcal{L}(\tilde{\mathbf{n}}, \tilde{\boldsymbol{\alpha}}, \tilde{\rho}) = \mathcal{L}_s(\tilde{\mathbf{n}}) + \lambda_p \sum_j \mathcal{L}_p^j(\tilde{\mathbf{n}}, \tilde{\boldsymbol{\alpha}}, \tilde{\rho}) + \lambda_c \mathcal{L}_c(\tilde{\boldsymbol{\alpha}}). \quad (3.14)$$

We set the weight  $\lambda_p$  to 10 and  $\lambda_c$  to 50 based on the validation dataset.

**Data Augmentation and Training.** To improve the robustness of our network, we apply a series of data augmentations to our captured OLATs to simulate a variety of different visible light conditions. Specifically, our training uses a combination of: evenly-lit RGB inputs obtained by adding together all of the OLAT images; inputs with strong shadows by selecting exactly one of the OLAT images; a mixture of two lights with different temperatures by applying randomly chosen color vectors to two randomly chosen OLAT images; low-light environments by adding Gaussian noise to a single OLAT; and saturated exposures by scaling and clipping a single OLAT. We sample evenly from these 5 lighting conditions during training. Further details on how these lighting conditions are simulated are provided in the supplementary material.

We train the network using the Adam optimizer [70] for 30K iterations, with a learning rate of  $10^{-3}$  and a batch size of 8. Training takes 12 hours with 4 Tesla V100 GPUs.



### 3.2.4 Dataset

Shown in Figure 3.8, our setup combines a 7.0MP RGB camera that operates at 66.67 fps with a stereo pair of 2.8MP NIR cameras that operate at 150 fps. The RGB camera and one of the NIR cameras are co-located using a plate beamsplitter and a light trap. The RGB and NIR cameras have a linear photometric response and we downsample all of the images by a factor of 2 in each dimension and take a central crop that covers the face at a resolution of  $960 \times 768$ .

Visible spectrum lighting is provided by 4 wide-angle LED spotlights placed at the corners of a roughly  $1.5m \times 0.8m$  (width x height) rectangle surrounding the cameras located approximately 1.1m from the subject. NIR lighting is provided by 5 NIR spotlights, one adjacent to each of the visible lights, and a flash LED light located near the reference NIR camera to produce the “dark flash” input. These NIR light sources are temporally interleaved with projectors that emit NIR dot speckle patterns to assist stereo matching [128]. A microcontroller orchestrates triggering the lights and cameras to ensure that at any time only one visible light source and one NIR light source is active. All light sources are calibrated for position and intensity and treated geometrically as point light sources. The light intensity term  $L$  in Equation 3.10 accounts for these calibrated colors. Note that the NIR and visible light sources are not colocated and so slightly different values of  $\mathbf{l}$  are used in Equation 3.10 between those two conditions.

The image acquisition rate is limited by the RGB camera’s framerate and the total light output, but is fast enough for us to record video sequences of people who are gesturing and moving slowly. We compute optical flow [171] between consecutive frames captured under the same lighting condition to correct for the small amount of scene motion that occurs

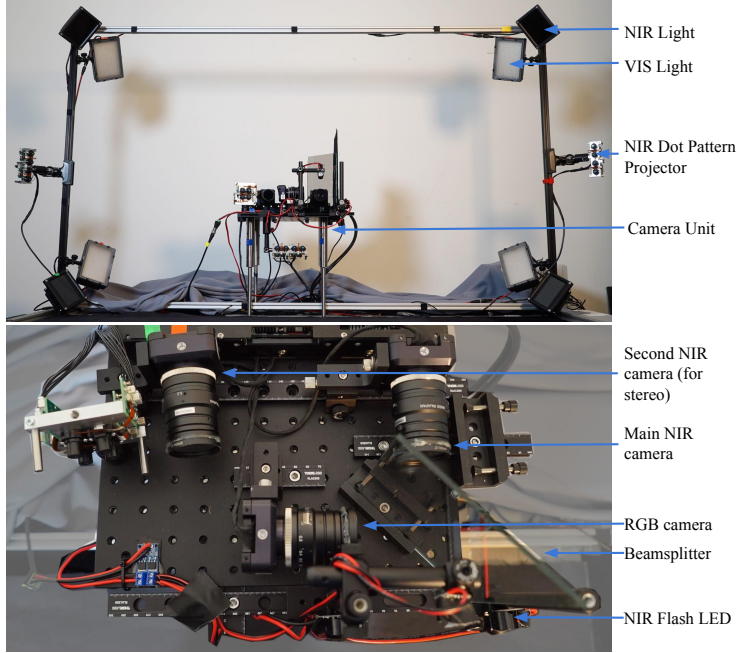


Figure 3.8: Our hardware setup consists of controllable NIR and visible spectrum light sources, an RGB camera, a stereo pair of NIR cameras, and two NIR dot projectors. One of the NIR cameras and the RGB camera are aligned with a beamsplitter and all of these components are triggered electronically to record the types of images shown in Figure 3.6.

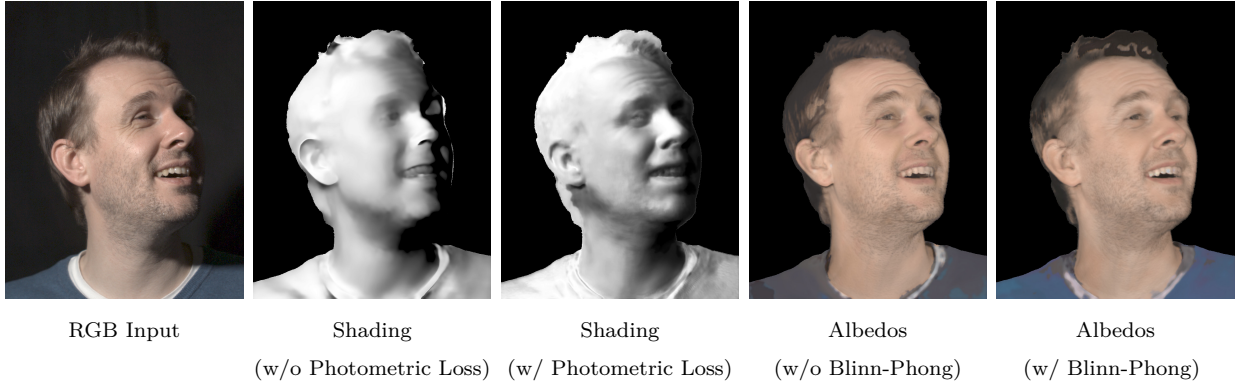


Figure 3.9: Impact of the photometric loss term in our training procedure and the Blinn-Phong BRDF in our image formation model, respectively. When trained without photometric loss, our network learns to output the stereo normals, which lack fine-scale details. This has a fairly small effect on the error measures in Table 3.3, but is perceptually significant as seen in these “n dot l” shading renderings. Our full image formation model, which includes a Blinn-Phong specular term, produces more accurate albedos across the face than using a Lambertian model alone.

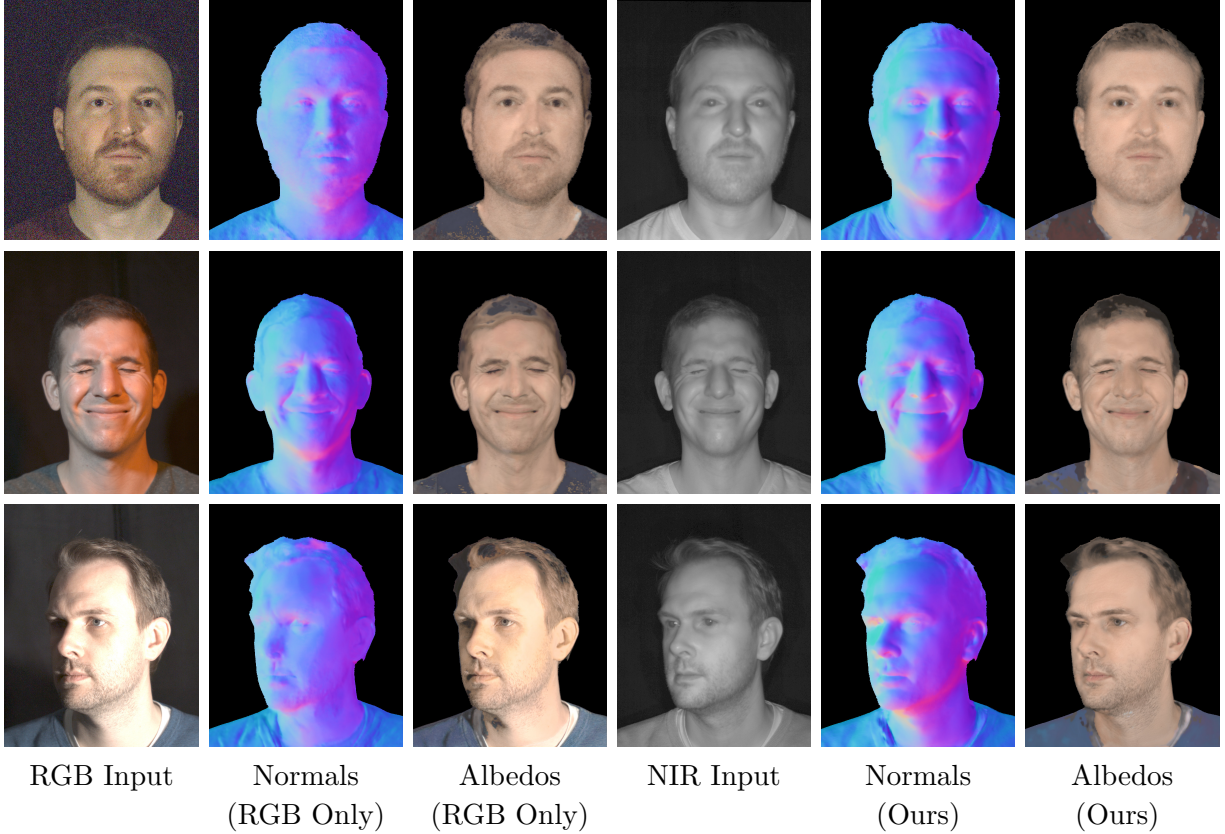


Figure 3.10: Comparison of our network to a modified version that takes only a single RGB image (“RGB Only”) as input. Example results for three common challenging lighting conditions. Top to bottom: low light / noisy inputs; mixed light colors; harsh directional lighting with saturated intensities. The “RGB only” network struggles to produce stable normal and reflectance estimates from these inputs in contrast to our method.

within a single round of exposures. Since the RGB and reference stereo NIR camera are co-located, we can generate pixel-aligned RGB, NIR, and depth images using scene-independent precomputed image warps.

Each recording in our dataset is 10 seconds long and contains 166 sets of frames. We recorded 9 unique subjects, with between 5 and 10 sessions per subject, for a total of 61 recordings. We used recordings of 6 of the subjects for training and tested on recordings of the other 3.

### 3.2.5 Evaluation

To the best of our knowledge, our method is the first technique for estimating surface normals and RGB albedos from an RGB+NIR image. We demonstrate the value of utilizing NIR inputs by comparing our method to two state-of-the-art RGB-only face normal estimation methods [124, 145] as well as an RGB-only variant of our own method. We also perform several other ablation studies to measure the impact of key design decisions. To illustrate the performance of our method in lighting conditions that do not lie in the span of our captured OLAT images, we also show qualitative results (Figure 3.5) on a real sequence captured while casually moving a handheld light source around the scene. Note that ground-truth normal maps are not available for this sequence. Finally, we present two applications of our technique. None of the subjects shown in our results are in our training set.

**Comparisons and Ablation Studies.** In our evaluations we consider five different visible lighting conditions: harsh lighting that produces strong cast shadows; a mixture of lights with different color temperatures; saturated/overexposed intensities; low-light conditions that produce noisy inputs; and a “well lit” condition that achieves largely shadow-free and well exposed inputs. In lieu of ground truth geometry for quantitative assessments, we construct a baseline using the technique of Nehab et al. [123] to refine our stereo depth maps according to normals computed by applying Lambertian photometric stereo to the RGB OLAT training images.

Table 3.3 reports the mean absolute angular errors in normal maps computed by two state-of-the-art RGB-based face normal estimation methods [124, 145] along with several variants of our network with different loss terms, image formation models, and inputs. Figures 3.9

	Well lit	Shadows	Mixed colors	Overexposure	Low light
SfSNet [145]	14.10	18.32	-	-	-
Nestmeyer et al.[124]	14.82	17.52	15.87	21.85	25.56
Ours (No Stereo Loss)	12.80	12.78	12.78	12.82	12.81
Ours (No NIR Photometric Loss)	12.64	12.66	12.64	12.69	12.75
Ours (No Photometric Loss)	12.77	12.77	12.81	12.79	12.77
Ours (No Specular Component)	12.44	12.43	12.44	12.51	12.47
Ours (No RGB Input)	12.54	12.54	12.54	12.54	12.54
Ours (No NIR Input)	13.13	15.19	16.43	19.82	19.39
<b>Ours</b>	<b>12.08</b>	<b>12.06</b>	<b>12.06</b>	<b>12.14</b>	<b>12.10</b>

Table 3.3: Mean absolute angular error in degrees of normal maps computed with modified versions of our full network. Results are reported for the five lighting conditions described in Section 3.2.5.

and 3.10 show examples of the perceptual impact of some of these design decisions.

We first compare to SfSNet [145] and Nestmeyer et al [124]. As shown in Table 3.3, our method outperforms both techniques even in the well lit condition and without using the NIR input, which we attribute to our novel training strategy that combines shape information from complementary stereo and photometric signals. More importantly, in challenging lighting conditions, the benefit of our method becomes far more significant as the additional information provided by the NIR input is crucial in these circumstances. Note that SfSNet [145] uses a self-reconstruction loss that we found could not handle inputs with mixed color casts, saturated intensities, or a significant amount of noise and so it fails to produce plausible outputs in these cases (omitted from Table 3.3).

We next show the impact of our design choices. As expected, using both stereo and photometric loss terms during training outperforms using either one alone. We consider two types of photometric loss - one computed on only the RGB training images (“No NIR Photometric Loss” in Table 3.3) and the second computed on both the NIR and RGB training images (“Full Method”). As illustrated in the shading images in Figure 3.9, including the photometric

loss enables estimating fine geometric details that are not captured in the stereo depth maps.

Including the Blinn-Phong BRDF in our image formation model improves the accuracy of the normals and diffuse albedo maps. It results in a modest improvement in the quantitative errors in Table 3.3, and it produces more uniform diffuse albedo maps with fewer artifacts (Figure 3.9). We attribute this to the fact that this richer image formation model is better able to explain the observed intensities. We also found that including this BRDF in our model enables reconstructing the glossy appearance of skin (Figure 3.12).

Including the NIR input image improves accuracy across the board, especially in poor visible lighting conditions (Table 3.3). The benefit of the RGB input is comparatively smaller, but making it available to the network enables estimating visible spectrum reflectance data, which is a requirement for many downstream applications such as lighting adjustment (Figure 3.12). Figure 3.10 illustrates the perceptual impact of including the NIR input in different lighting conditions. For these comparisons we modified our network to take only a single RGB image as input (“RGB Only”). The network architecture was otherwise unchanged, and we applied the same training procedure described in Section 3.2.3. Note how the performance of this “RGB Only” network significantly degrades in challenging conditions, while our method is far more robust to these conditions due to the more stable NIR input. It’s particularly noteworthy how well our method is able to reconstruct plausible diffuse albedos even for highly saturated RGB input images (bottom row of Figure 3.10).

**Application: Stereo Refinement.** Stereo methods excel at measuring coarse geometry, but often struggle to recover fine-scale surface details. This can be overcome by refining stereo depths according to accurate high-resolution normals typically estimated with a photometric

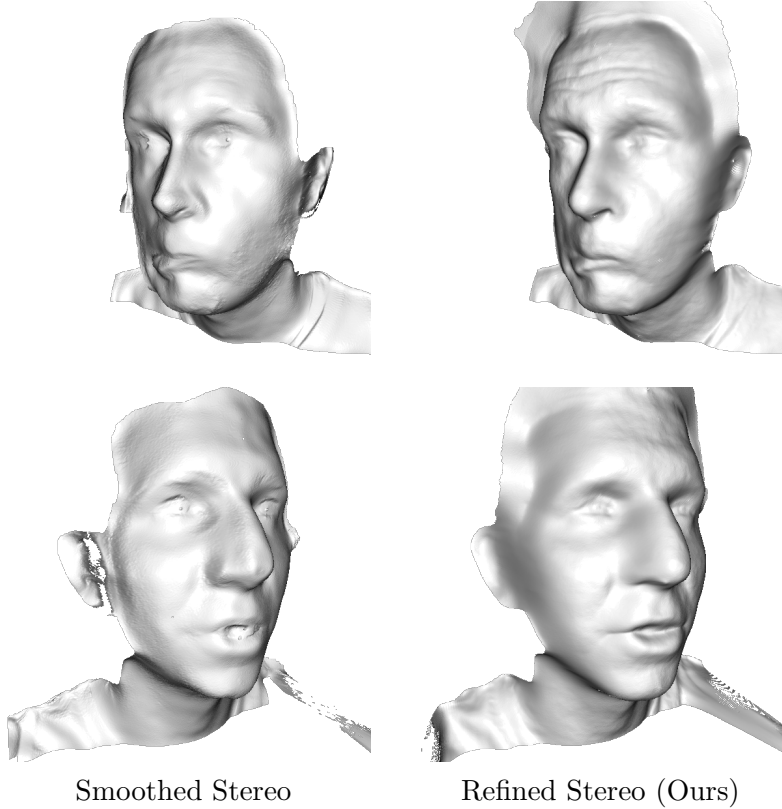


Figure 3.11: Stereo methods often struggle to recover fine-scale surface details. *Left:* Applying a guided bilateral filter to raw stereo depths yields a smoother surface but with distorted features (e.g. the nose is reduced and skin wrinkles are missing). *Right:* We use the method of Nehab et al. [123] to compute a refined surface according to normals estimated with our method. Note how details are better preserved around the eyes, nose, and mouth, along with fine wrinkles and creases.

approach [123]. We evaluate using the normals produced by our method to refine depth measurements produced by an NIR space-time stereo algorithm [128] (Figure 3.11). In comparison to using a standard bilateral filter to smooth the stereo depths, refining them using our normals gives much higher quality reconstructions, most notably around the mouth, nose, and eyes and better recovery of fine wrinkles and creases in the skin. As our method works with a single NIR image it would be straightforward to integrate it into many existing stereo pipelines.

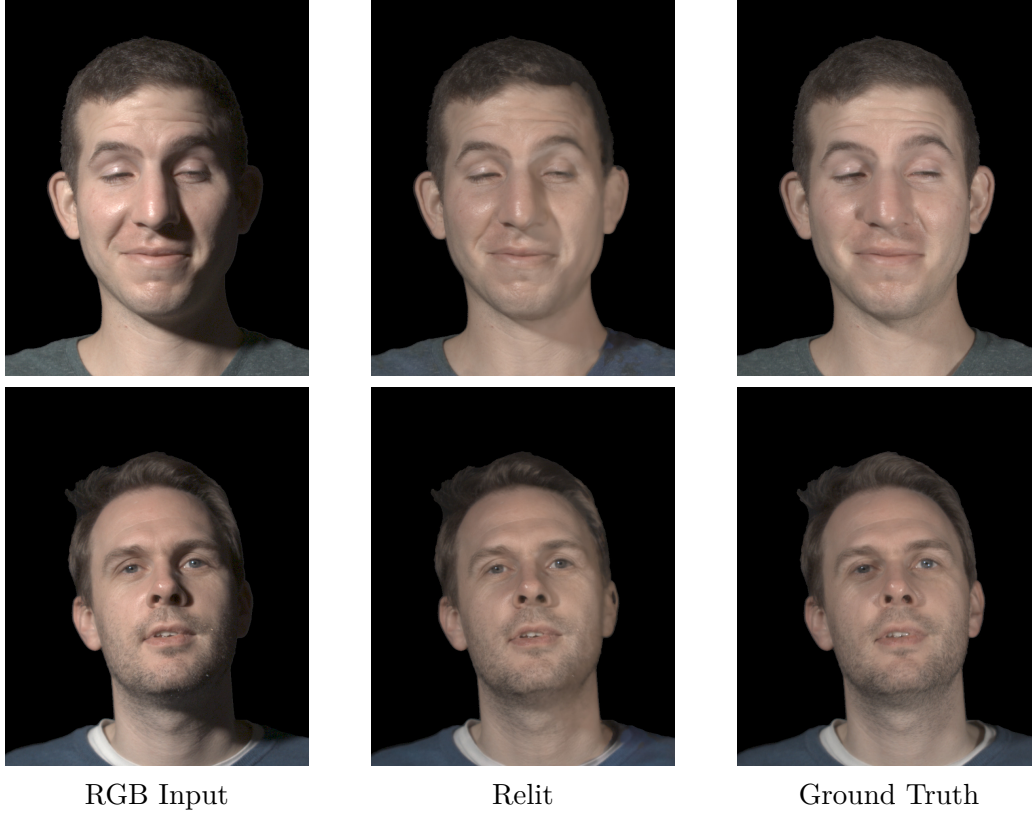


Figure 3.12: Our method can be used to simulate adding lights to a scene to fill in shadows.

**Application: Lighting Adjustment.** We also explored using our approach to digitally improve the lighting in a portrait. Specifically, we evaluated adding a virtual fill light to brighten shadowed parts of the face (Figure 3.12). We used normal and reflectance maps estimated by our method to render the contribution of a virtual point light located within view of the shadowed region, and then combined this with the original RGB image. Our model enables a convincing effect, even producing realistic specular highlights along the nasolabial folds and the tip of the nose.

### 3.2.6 Discussion

We have presented a dark flash normal camera that is capable of estimating high-quality normal and reflectance maps from a single RGB+NIR input image that can be recorded



in a single exposure without distracting the subject. Notably, our training is supervised by two indirect and complementary signals: one from stereo triangulation and the other from photometric cues in RGB and NIR. A key benefit of our method over prior work is its robustness. It performs well even in challenging lighting conditions that are commonly encountered in casual photography such as harsh shadows, saturated pixels, and in very low light environments. Our method could easily be integrated into existing smartphone camera hardware designs and software pipelines to enable a range of applications from refining the output of an auxiliary depth camera to improving the lighting of faces in still images and streaming video.

## Chapter 4

# Probabilistic scene map estimation for modular inference

Scene map estimation methods—that decode scene properties from a single or multiple photographs—have achieved surprising success through the use of deep neural networks [21, 35, 39, 43, 49, 79, 82, 95, 145, 146, 161, 183]. This success confirms that even a single photograph contains considerable information about different properties of the scene, such as geometry and reflectance. However, these computational photography tasks are extremely challenging given their ill-posed nature. Therefore, estimations from only photographs are far from being precisely accurate. Fortunately, many practical systems are able to rely on other (yet also imperfect) sources of scene information—successive frames for image restoration, limited measurements from depth sensors for monocular depth estimation, interactive user guidance for 3D mesh reconstruction, etc. And so, it is desirable to combine these other sources with the input photographs to extract scene map estimations that are more accurate than possible from one source alone.

Although the cues in images are useful for augmenting other sources of scene properties, the same isn't true for scene map *estimators* that simply output a scene map, a form which can not be directly combined with additional knowledge of the scene. Instead, researchers have treated scene map estimation using different combinations of cues as different applications in their own right (*e.g.*, depth estimation from sparse measurements [29]), and solved each by learning separate estimators that take their corresponding set of cues, in addition to the color image, as input. This requires, for each application, determining the types of inputs that will be available, constructing a corresponding training set, choosing an appropriate network architecture, and then training that application-specific network—a process that is redundant and often onerous.

In this chapter, we seek to train a scene map estimation neural network to output a rich representation of our belief and uncertainty on the scene property given the input image. Rather than producing a “best guess” of the scene map, we propose to output a *probability distribution*. Such output cannot simply be a per-pixel probability distribution, as we must characterize the spatial dependencies of different pixels in the scene map estimation. The proposed network can be trained in an application-agnostic way with image and scene map pairs, but can be utilized for inference in different applications and combined with different external information about the scene property.

Specifically, we focus on the task of monocular depth estimation. We train a conditional VAE [73] to output multiple plausible depth samples independently for individual overlapping patches, and form the density as a sample approximation from all samples and patches. Our distributional output is versatile enough to enable a diverse variety of applications as illustrated in Figure 4.1. It is useful even in the purely monocular setting—when only a single image is available—and can be used to produce accurate depth predictions, a measure of confidence in these predictions, as well as estimates of relative ordering of pairs of scene points.

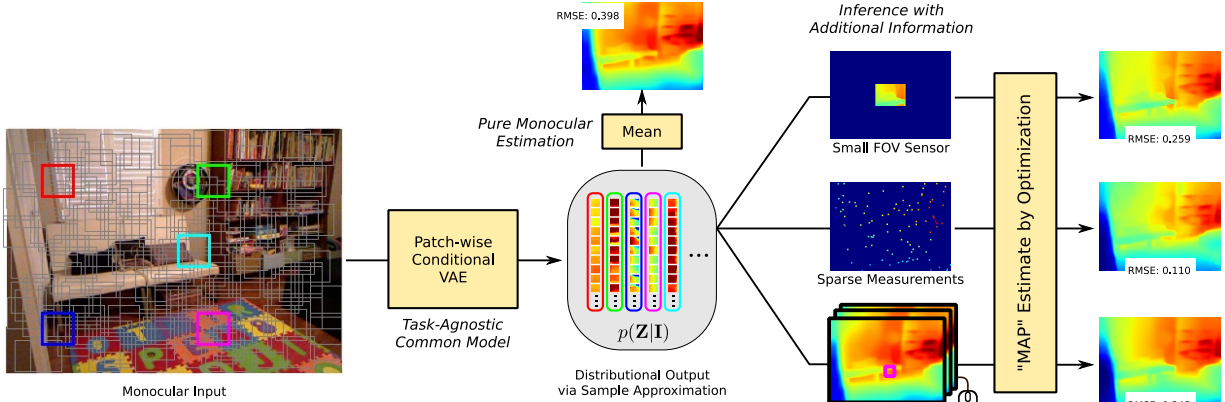


Figure 4.1: Overview of our approach. Given an input color image, we use a common task-agnostic network to output a joint probability distribution  $p(\mathbf{Z}|\mathbf{I})$  over the depth map—formed as a sample approximation using outputs of a conditional VAE that generates plausible estimates for depth in overlapping patches. The mean of this distribution represents a standard monocular depth estimate, but the distribution itself can be used to solve a variety of inference tasks in different application settings—including leveraging additional depth cues to yield improved estimates. All these applications are enabled by a common model, *that is trained only once*.

More importantly, it is also able to incorporate additional information to produce improved depth estimates in diverse application settings: producing multiple depth maps for user selection, incorporating user annotation of erroneous regions, incorporating a small number of depth measurements—along a single line, within a smaller field of view, at random as well as regular sparse locations—and selecting the optimal locations for these measurements. Crucially, all of these applications are enabled by the same network model that is trained only once, while achieving accuracy comparable to state-of-the-art methods that rely on separate task-specific models.

This chapter is structured as follows. Section 4.1 presents a review of related works in depth estimation. We describe our approach to output a joint distribution over depth and its result in the monocular setting, i.e., when only a single image is available in Section 4.2. We then

present our optimization method to combine this joint distribution with other sources of depth information in Section 4.3.

## 4.1 Related work

**Monocular Depth Estimation.** First attempted by Saxena *et al.* [142], early work in estimating scene depth from a single color image relied on hand-crafted features [81, 133, 143, 148], use of graphical models [106, 143, 195], and databases of exemplars [68, 75]. More recently, Eigen *et al.* [40] showed that, given a large enough database of image-depth pairs [151], convolutional neural networks could be trained to achieve significantly more reliable depth estimates. Since then, there have been steady gains in accuracy through the development of improved neural network-based methods [21, 39, 49, 60, 84, 91, 103, 138, 163, 191], as well as strategies for unsupervised and semi-supervised learning [26, 50, 80]. Beyond estimating absolute depth, some works have also looked at pairwise ordinal depth relations between pair of points in the scene from a input color image [26, 197].

**Probabilistic Outputs.** Monocular depth estimators commonly output a single estimate of the depth value at each pixel, hindering their use in different estimation settings. Some existing methods do produce distributional outputs, but as per-pixel variance maps [60, 69] or per-pixel probability distributions [101]. Note that depth values at different locations are not statistically independent, i.e., different values at different locations may be plausible independently, but not in combination. Thus, per-pixel distributions provide only a limited characterization that, while useful in some applications, can not be used more generally, *e.g.*, to spatially propagate information from sparse measurements.

Beyond per-pixel distributions, Chakrabarti *et al.* [21] train a network to produce independent distributions for different local depth derivatives. They describe a method to use these derivative distributions to generate a better estimate of global depth, but do not provide a way to solve other tasks. Also, since their network output is restricted to uni-variate distributions for hand-chosen derivatives, it can not express the general spatial dependencies in a joint distribution over depth that we seek to encode for inference.

**Depth from Partial Measurement.** Since making dense depth measurements is slow and expensive, it is useful to be able to recover a high-quality dense depth map from a small number of direct measurements by exploiting the monocular cues in a color image. A popular way of combining color information with partial measurements is by requiring color and depth edges to co-occur: this approach is often successful for “depth inpainting”, i.e., filling in gaps of missing measurements in a depth map (common in measurements from structured light sensors). A notable and commonly-used example is the colorization method of Levin *et al.* [88]. Other methods along this line include [38, 61, 104, 105, 117], while Zhang and Funkhouser [189] used a neural network to predict normals and occlusion boundaries to aid inpainting.

However, when working with a very small number of measurements, the task is significantly more challenging (see discussion in [29]) and requires relying more heavily on the monocular cue. In this regime, the solution has been to train a network that takes the color image and the provided sparse samples as input. Various works have adopted this approach for measurements along a single horizontal line from a line sensor [98], random sparse measurements [65, 111, 149, 158], and sub-sampled measurements on a regular grid [29, 54, 93]. Note that several of these methods also train separate networks even for different settings of the same application, such as for different sparsity levels [111] and different resolution grids [29].

An exception here is the depth completion method of Wang *et al.* [162] who use a pre-trained monocular depth network, and provide a way to improve its monocular predictions when given sparse depth measurements. They iteratively back-propagate errors between measurements and the network output to update activations of an intermediate layer (but not the network weights), leading to an improved depth map output. Thus, their method uses the monocular network’s output as an initialization, and its internal representation as a structured way to spatially propagate measurement information. In contrast, our method outputs an explicit probabilistic representation which can be used for depth completion as well as for other inference tasks, and as our experiments show, yields more accurate results.

**Networks for Generating Samples.** In this work, we form a conditional joint distribution of depth values by training our network to generate samples of multiple plausible depth values. In particular, we follow the approach of [73] to train a conditional VAE and use its outputs to form a sample approximation to the joint distribution. Note that instead of generating samples of a global map (like in [73]), we train the VAE to produce samples for individual overlapping patches independently. We also conduct ablation experiments using a conditional GAN [53, 121] to produce these samples, and while the VAE formulation performs better, our results with the GAN are also reasonable. This suggests our approach is able to exploit any neural network-based method for generating conditional samples, and can benefit from future advances in this direction.

## 4.2 Probabilistic monocular depth

### 4.2.1 Proposed method

Given the RGB image  $\mathbf{I}$  of a scene, our goal is to reason about its corresponding depth map  $\mathbf{Z} \in \mathbb{R}^N$ , represented as a vector containing depth values for all  $N$  pixels in the image. Rather than predict a single estimate for  $\mathbf{Z}$ , we seek to output a *distribution*  $p(\mathbf{Z}|\mathbf{I})$ , to more generally characterize depth information and ambiguity present in the image. In this section, we describe our approach for generating this distributional output, and equally importantly, for exploiting it for inference in various applications.

We form the distribution  $p(\mathbf{Z}|\mathbf{I})$  as a product of functions defined on individual overlapping patches as

$$p(\mathbf{Z}|\mathbf{I}) \propto \prod_i \psi_i(\mathcal{P}_i \mathbf{Z}|\mathbf{I}), \quad (4.1)$$

where  $\psi_i(\cdot)$  is a potential function for the  $i^{th}$  patch, and  $\mathcal{P}_i$  a sparse matrix that crops out that patch from  $\mathbf{Z}$  (for patches of size  $K \times K$ , each  $\mathcal{P}_i$  is a  $K^2 \times N$  matrix). Note that this is a Markov Random Field with  $K \times K$  patches as the maximal cliques, and since these patches overlap, depth values at all pixels—including those that do not lie in the same patch—are statistically inter-dependent.

**Generating Samples.** To form the per-patch potentials  $\psi_i(\cdot)$ , we train a network that produces samples of depth given the image input, and run it multiple times during inference to generate multiple plausible samples. A crucial aspect of this network is that, instead of sampling the global depth map, it generates separate samples independently for the depth  $\mathcal{P}_i \mathbf{Z}$  of every patch  $i$ . This ensures that depth values within each sample represent



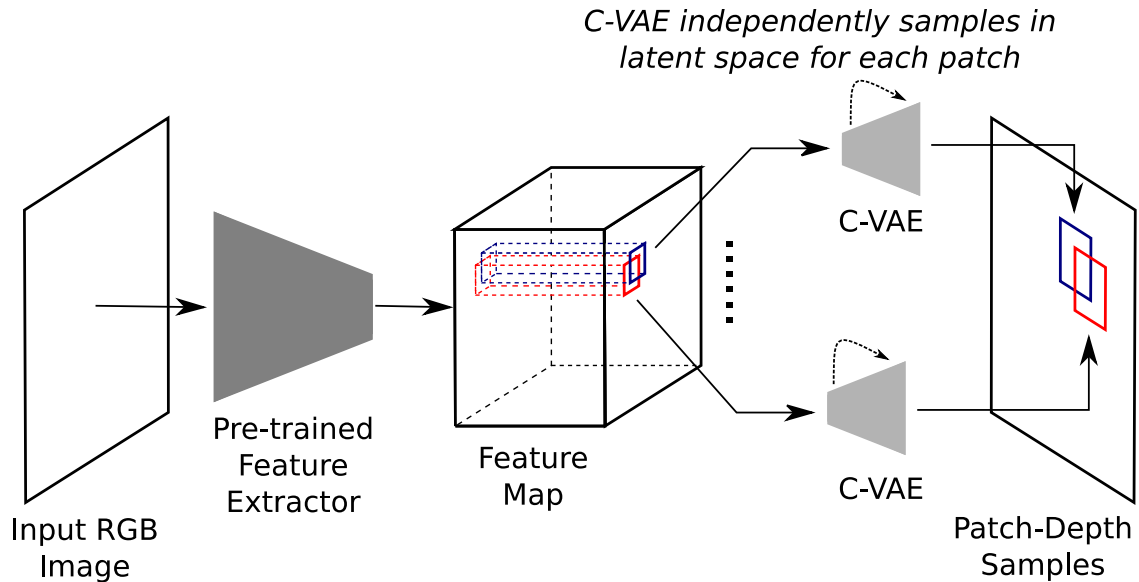


Figure 4.2: Generating samples with a conditional VAE. Our network generates samples for depth independently in each overlapping patch, and we run it multiple times to generate multiple plausible samples per-patch. The input to the VAE comes from pre-trained feature extraction layers from a state-of-the-art monocular model [49]. Samples generated for different patches (including those that overlap) are kept statistically independent—after conditioning on the image—by using separate per-patch latent vectors.

a plausible estimate for the corresponding patch, but that samples of different patches are conditionally independent given the image. Limiting the dimensionality of each sample allows us to approximate the per-patch potential  $\psi_i(\cdot)$  with a reasonable number of samples, while enforcing independence between samples of different patches ensures that the overall distribution  $p(\mathbf{Z}|\mathbf{I})$  in (4.1) sufficiently captures the global ambiguity in depth.

We adopt the conditional VAE framework proposed in [73] for generating samples—that features a “prior-net” to predict distribution over values of a latent vector from the image, with an encoder-decoder network that predicts depth values from the image and a sample from this latent distribution. To reduce complexity, we bootstrap our network by taking a pre-trained state-of-the-art monocular depth estimation network (DORN [49]), removing the

last two convolution layers, and treating the remaining layers as a “feature extractor”. These features, rather than the image itself, are provided as input to the conditional VAE.

We achieve patch independent sampling by having a separate latent vector for each patch. We set up the architecture of the decoder in the encoder-decoder network to produce an estimate of the depth of each overlapping patch using only its own latent vector, and not those of overlapping patches. The prior-net is also setup to predict separate distributions for the latent vector of each patch (as is the posterior-net during training). At test time, we draw multiple samples independently from the latent space for each patch, which the encoder-decoder network uses to generate correspondingly independent per-patch depth samples.

**Sample Approximation.** Next, given a set  $\mathcal{S}_i$  of samples  $\{\mathbf{x}_i^s\}$  for each patch  $i$ , we define its potential  $\psi_i(\cdot)$  as

$$\psi_i(\mathcal{P}_i \mathbf{Z} | \mathbf{I}) = \frac{1}{|\mathcal{S}_i|} \sum_{\mathbf{x}_i \in \mathcal{S}_i} \exp \left( -\frac{\|\mathcal{P}_i \mathbf{Z} - \mathbf{x}_i\|^2}{2h^2} \right). \quad (4.2)$$

This can be interpreted as forming a kernel density estimate from the depth samples in  $\mathcal{S}_i$  using a Gaussian kernel, where the Gaussian bandwidth  $h$  is a scalar hyper-parameter<sup>3</sup>.

Unlike independent per-pixel [60, 69, 101] or per-derivative [21] distributions, the samples  $\{\mathcal{S}_i\}$  enable the patch potentials  $\psi_i(\cdot)$  to express complex spatial dependencies between depth values in local regions. Moreover, our joint distribution  $p(\mathbf{Z} | \mathbf{I})$  is defined in terms of overlapping patches, and thus models dependencies across the entire depth map. During inference, this enables information propagation across the entire scene, and reasoning about the global plausibility of scene depth estimates.

---

<sup>3</sup>While  $h$  can be estimated based on the variance between  $\mathbf{x}_i$  and true patch depths, as we will see, its actual value is often not needed as it is factored into other manually-set, task-specific parameters.

### 4.2.2 Monocular inference tasks

Note that the distribution  $p(\mathbf{Z}|\mathbf{I})$  can be used to recover a monocular depth map estimate as the mean over  $p(\mathbf{Z}|\mathbf{I})$  by computing the average estimate of depth at each pixel from all samples from all patches that include that pixel. But our distributional output is also versatile and can be used to perform general monocular inference tasks, not just estimate per-pixel depth. We describe two such applications below.

**Confidence-guided Sampling.** We can use  $p(\mathbf{Z}|\mathbf{I})$  to compute a per-pixel variance map, as the variance of each pixel’s depth value across patches and samples in  $\{\mathcal{S}_i\}$  (which differs from the actual variance under  $p(\mathbf{Z}|\mathbf{I})$  by a constant  $h^2$ ). This gives us spatial map of the relative monocular ambiguity in depth at different locations. When seeking to estimate depth from arbitrary sparse measurements, we can use this map to select where to make measurements (assuming the depth sensor provides such control). Specifically, given a budget on the total number of measurements, we propose choosing an optimal set of measurement points as local maxima of the variance map.

**Pair-wise Depth.** A useful monocular depth inference task, introduced in [197], is to predict the ordinal relative depth of pairs of nearby points in the scene: whether the points are at similar depths (within some threshold), and if not, which point is nearer. We use our distributional output to solve this task, by looking at the relative depth in all samples in all patches that contain a pair of queried points, outputting the ordinal relation that is most frequent. We find this leads to more accurate ordinal estimates, in comparison to simply using the ordering of the individual depth value pairs in a monocular depth map estimate (as done in [26, 197]).

### 4.2.3 Experimental results

We now evaluate our approach on the NYUv2 dataset [151] by training a common task-agnostic distributional monocular model and applying it to solve monocular inference tasks.

**Preliminaries.** We use raw frames from scenes in the official train split for NYUv2 [151] to construct train and val sets, and report performance on the official test set. We use feature extraction layers from a pre-trained DORN model [49], and since it operates on inputs and outputs rescaled to a lower resolution (to  $257 \times 353$  from  $640 \times 480$ ), we do the same for our VAE. However, our outputs are rescaled back to the original full resolution to compute errors. Input depth measurements, if any, are also provided at full resolution. We use overlapping patches of size  $33 \times 33$  with stride four, and generate 100 samples per-patch to construct  $\{\mathcal{S}_i\}$ . Generating samples takes 5.7s on a 1080Ti GPU for each image, while inference from these samples is faster.

**Performance on Monocular Depth Estimation.** We evaluate depth estimation using our model for monocular depth estimation, and report performance in terms of standard error metrics on the official NYUv2 test set (see [39])<sup>4</sup> in Table 4.1. Although monocular depth estimation is not the end goal of our versatile probabilistic estimation, our method still perform well in the monocular setting—outperforming the DORN [49] whose features it uses. Figure 4.3 shows example depth reconstructions by our method.

**Performance on Confidence-guided Sampling.** In Table 4.2, we report results for making sparse depth measurements guided by the color image using our approach for different

---

<sup>4</sup>Some papers interpret RMSE as mean of per-image RMSE values. We report the standard definition as rms, and this per-image version as m-rms.

Setting	Method	lower is better			higher is better		
		rms	m-rms	rel	$\delta_1$	$\delta_2$	$\delta_3$
<b>Monocular Depth Estimation</b>							
	Lee [85]	0.538	0.470	0.131	83.7	97.1	<b>99.4</b>
	DORN [49]	0.545	0.462	<b>0.114</b>	85.8	96.2	98.7
	<i>Ours</i>	<b>0.512</b>	<b>0.433</b>	0.116	<b>86.1</b>	<b>96.9</b>	99.1

Table 4.1: Results our probabilistic output for monocular depth estimation on the NYUv2 test set. Methods that we compare to are specifically proposed for monocular depth estimation.

budgets on the number of measurements. Our guided measurements lead to better dense depth estimates than those at random locations (given measurements, we use our depth estimation algorithm described in Section 4.3.2 in both cases).

Measurements	20	50	100	200
Random	0.359	0.320	0.279	0.246
<i>Guided</i>	<b>0.331</b>	<b>0.286</b>	<b>0.253</b>	<b>0.227</b>

Table 4.2: RMS error for depth estimation from different numbers of sparse measurements, when making measurements at random locations vs. with guidance from our distribution. Given the measurements, we use our depth estimation algorithm described in Section 4.3.2 in both cases

**Performance on Pair-wise Depth.** We evaluate using our distribution to predict pairwise depth ordering in Table 4.3, comparing it to three methods that specifically target this task: [26, 170, 197]. Results are reported in terms of the WKDR error metrics, on a standard set of point pairs on the NYUv2 test set (see [197]). We find that using our method leads to better predictions than from these methods, and that using our distributional output is crucial—since the accuracy of simply using the orderings from our monocular mean estimate is much lower.

Method	WKDR	WKDR <sup>=</sup>	WKDR <sup>≠</sup>
Zoran [197]	43.5%	44.2%	41.4%
Chen [26]	28.3%	30.6%	28.6%
Xian [170]	29.1%	29.5%	29.7%
Ours: mean	30.2%	29.9%	30.5%
<i>Ours (distribution)</i>	<b>27.1%</b>	<b>26.0%</b>	<b>27.8%</b>

Table 4.3: Error rates for pairwise ordinal depth ordering from our common model, compared to other methods that used accurate ordering as an objective during training. We also report baseline errors from predictions just based on our mean depth estimate.

Although our output distribution  $p(\mathbf{Z}|\mathbf{I})$  can be used to achieve state-of-the-art performance in several monocular applications, the real utility of our distributional output comes from enabling a variety of inference tasks, as we describe next.

## 4.3 Depth estimation with additional information

### 4.3.1 Proposed approach

In several applications, a system has access to additional sources beyond the monocular image that provide some partial information about depth. Our distributional output allows us to combine the monocular cue with these sources, and derive a more accurate scene depth estimate than possible from either source alone. Specifically, we assume the additional depth information is provided in the form of a cost  $C(\mathbf{Z})$ , and combine it with our distribution  $p(\mathbf{Z}|\mathbf{I})$  to derive a depth estimate  $\hat{\mathbf{Z}}$  as:

$$\begin{aligned}\hat{\mathbf{Z}} &= \arg \min_{\mathbf{Z}} -\log p(\mathbf{Z}|\mathbf{I}) + C(\mathbf{Z}), \\ \log p(\mathbf{Z}|\mathbf{I}) &= \sum_i \log \sum_{\mathbf{x}_i \in \mathcal{S}_i} \exp \left( -\frac{\|\mathcal{P}_i \mathbf{Z} - \mathbf{x}_i\|^2}{2h^2} \right).\end{aligned}\tag{4.3}$$

With some abuse of terminology, this can be thought of as computing the maximum a posteriori (MAP) estimate of  $\mathbf{Z}$ , where  $p(\mathbf{Z}|\mathbf{I})$  is the image-conditional “prior”, and  $C(\mathbf{Z})$  can be interpreted as a “likelihood” from the additional depth information source.

The log-likelihood of our distribution in (4.3) can be simplified with a standard approximation of replacing the summation over exponentials with a maximum (since  $\mathcal{P}_i \mathbf{Z}$  is high-dimensional, the largest term typically dominates):

$$\begin{aligned}\hat{\mathbf{Z}} &\approx \arg \min_{\mathbf{Z}} - \sum_i \log \max_{\mathbf{x}_i \in \mathcal{S}_i} \exp \left( -\frac{\|\mathcal{P}_i \mathbf{Z} - \mathbf{x}_i\|^2}{2h^2} \right) \\ &\quad + C(\mathbf{Z}) \\ &= \arg \min_{\mathbf{Z}} \min_{\{\mathbf{x}_i \in \mathcal{S}_i\}} \sum_i \|\mathcal{P}_i \mathbf{Z} - \mathbf{x}_i\|^2 + 2h^2 C(\mathbf{Z}).\end{aligned}\tag{4.4}$$

Note that this expression now involves a minimization over both  $\mathbf{Z}$  and selections of samples  $\mathbf{x}_i \in \mathcal{S}_i$  for every patch.

We will use two forms of the external cost  $C(\mathbf{Z})$  to encode available information in various applications. The first is simply a generic global cost that we denote by  $C^G(\mathbf{Z})$ , and the other is one that can be expressed as a summation over the depth values of individual patches  $\sum_i C_i(\mathcal{P}_i \mathbf{Z})$ . Including both these possible forms in (4.4), we arrive at the following optimization task:

$$\min_{\mathbf{Z}} \min_{\{\mathbf{x}_i \in \mathcal{S}_i\}} \sum_i \|\mathcal{P}_i \mathbf{Z} - \mathbf{x}_i\|^2 + \underbrace{\sum_i C_i(\mathbf{x}_i) + C^G(\mathbf{Z})}_{\text{Possible forms of } C(\mathbf{Z})}, \quad (4.5)$$

where the factor  $2h^2$  is absorbed in the definitions of the costs, and the per-patch costs  $C_i(\mathcal{P}_i \mathbf{Z})$  are approximated as  $C_i(\mathbf{x}_i)$  to act on samples instead of crops of  $\mathbf{Z}$  (we assume this will roughly be equivalent at convergence).

We use a simple iterative algorithm to carry out this optimization. The global depth  $\mathbf{Z}$  is initialized to the mean per-pixel depth from  $p(\mathbf{Z}|\mathbf{I})$ , and the following updates are applied alternately to  $\{\mathbf{x}_i\}$  and  $\mathbf{Z}$  till convergence:

$$\mathbf{x}_i \leftarrow \arg \min_{\mathbf{x}_i \in \mathcal{S}_i} \|\mathcal{P}_i \mathbf{Z} - \mathbf{x}_i\|^2 + C_i(\mathbf{x}_i), \quad \forall i. \quad (4.6)$$

$$\mathbf{Z} \leftarrow \arg \min_{\mathbf{Z}} \|\mathcal{P}_i \mathbf{Z} - \mathbf{x}_i\|^2 + C^G(\mathbf{Z}). \quad (4.7)$$

The updates to patch estimates  $\mathbf{x}_i$  can be done independently, and in parallel, for different patches. The cost in (4.6) is the sum of the squared distance from corresponding crop  $\mathcal{P}_i \mathbf{Z}$  of the current global estimate, and the per-patch cost  $C_i(\cdot)$  when available. We can compute



these costs for all samples in  $\mathcal{S}_i$ , and select the one with the lowest cost. Note that the cost  $C_i(\cdot)$  on all samples need only be computed once at the start of optimization.

The update to the global map  $\mathbf{Z}$  in (4.7) depends on the form of the global cost  $C^G(\cdot)$ . If no such cost is present,  $\mathbf{Z}$  is given by simply the overlap-average of the currently selected samples  $\mathbf{x}_i$  for each patch. For applications that do feature a global cost, we find it sufficient to solve (4.7) by first initializing  $\mathbf{Z}$  to the overlap-average, and then carrying out a small number of gradient descent steps as

$$\mathbf{Z} \leftarrow \mathbf{Z} - \gamma \nabla_{\mathbf{Z}} C^G(\mathbf{Z}), \quad (4.8)$$

where the scalar step-size  $\gamma$  is a hyper-parameter.

### 4.3.2 Applications

We now discuss concrete examples of our inference approach by considering specific applications, and describe associated choices of the costs  $C^G(\cdot)$  and  $C_i(\cdot)$ .

**Dense Depth from Sparse Measurements.** We consider the task of estimating the depth map  $\mathbf{Z}$  when an input sparse set  $\mathbf{F}$  of depth measurements at isolated points in the scene is available, along with a color image. We use the measurements  $\mathbf{F}$  to define a global cost  $C^G(\cdot)$  in (4.5) as

$$C^G(\mathbf{Z}) = \lambda \|\mathbf{Z} \downarrow - \mathbf{F}\|^2, \quad (4.9)$$

where  $\downarrow$  represents sampling  $\mathbf{Z}$  at the measured locations. Based on this, we define the gradients to be applied in (4.8) for computing the global depth updates as

$$\nabla_{\mathbf{Z}} C^G(\mathbf{Z}) = \lambda (\mathbf{Z} \downarrow - \mathbf{F}) \uparrow, \quad (4.10)$$

where  $\uparrow$  represents the transpose of the sampling operation. Since both the weight  $\lambda$  and the step-size  $\gamma$  in (4.8) are hyper-parameters, we simply set  $\lambda = 1$ , and set the step-size  $\gamma$  (as well as number of gradient steps) based on a validation set.

We consider two kinds of sparse inputs. The first are at arbitrary random locations like in [65, 111, 149, 158, 162], where we use nearest neighbor interpolation for the transpose sampling operation  $\uparrow$  in (4.10). The other case is *depth up-sampling*, where measurements are on a regular lower-resolution grid. Given their regularity, we are able to use bi-linear interpolation for the transpose operation  $\uparrow$ .

**Depth Un-cropping.** We next consider applications where the available measurements are dense in a contiguous (but small) portion of the image—such as from a sensor with a smaller field-of-view (FOV), or along a single line [98]. In this case, we define  $\mathbf{F}$  and  $\mathbf{W}$  are set to measured values and one at measured locations, and zero elsewhere. We use these to define a per-patch cost  $C_i(\cdot)$  for use in (4.5) as

$$C_i(\mathbf{x}_i) = \lambda \|\mathcal{P}_i \mathbf{W} \circ (\mathcal{P}_i \mathbf{Z} - \mathcal{P}_i \mathbf{F})\|^2, \quad (4.11)$$

where the weight  $\lambda$  is determined on a validation set.

Depth estimates are often useful in interactive image editing and graphics applications. We consider a couple of settings where our estimation method can be used to include feedback from a user in the loop for improved depth accuracy.

**Diverse Estimates for User Selection.** We use Batra *et al.*'s approach [9] to derive multiple diverse global estimates  $\{\mathbf{Z}^1, \dots, \mathbf{Z}^M\}$  of the depth map  $\mathbf{Z}$  from our distribution  $p(\mathbf{Z}|\mathbf{I})$ , and propose presenting these as alternatives to the user. We set the first estimate

$\mathbf{Z}^1$  to our mean estimate, generate every subsequent estimate  $\mathbf{Z}^{m+1}$  by finding a mode using (4.5) with per-patch costs  $C_i(\cdot)$  defined as

$$C_i(\mathbf{x}_i) = -\lambda/m \sum_{m'=1}^m \|\mathcal{P}_i \mathbf{Z}^{m'} - \mathbf{x}_i\|^2. \quad (4.12)$$

This introduces a preference for samples that are different from corresponding patches in previous estimates, weighted by a scalar hyper-paramter  $\lambda$  (set on a validation set).

**Using Annotations of Erroneous Regions.** As a simple extension, we consider also getting annotations of regions with high error from the user, in each estimate  $\mathbf{Z}^m$ . Note that we only get the locations of these regions, not their correct depth values. Given this annotation, we define a mask  $\mathbf{W}^M$  that is one within the region and zero elsewhere, and now recover each  $\mathbf{Z}^{m+1}$ , with a modified cost  $C_i(\cdot)$ :

$$C_i(\mathbf{x}_i) = -\lambda/m \sum_{m'=1}^m \|(\mathcal{P}_i \mathbf{W}^{m'}) \circ (\mathcal{P}_i \mathbf{Z}^{m'} - \mathbf{x}_i)\|^2, \quad (4.13)$$

where  $\circ$  denotes element-wise multiplication, and the masks focuses the cost on regions marked as erroneous.

### 4.3.3 Experimental results

We now evaluate our approach on the same NYUv2 dataset [151] by applying the common task-agnostic distributional monocular model trained in Section 4.2.3 to solve a diverse range of inference tasks in various application settings.

The performance of our proposed method for the different depth completion and user guided applications described in Section 4.3.2 are reported in Table 4.4 and Table 4.5.

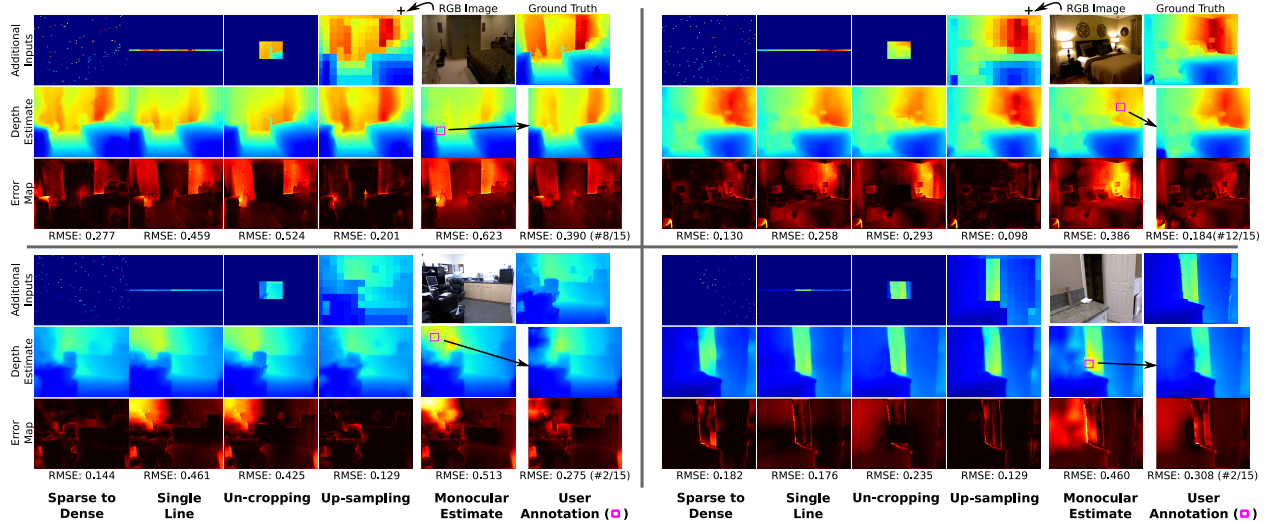


Figure 4.3: Example depth estimates for different applications. We show outputs from our method for both the pure monocular setting, as well as the improved estimates we obtain combining our distributional output with additional depth information—such as different kinds of partial measurements, and user guidance with annotation and selection.

Compare to our monocular results in Table 4.1, our approach is able to improve upon its monocular estimate with different available depth cues in the various applications. We find sparse measurements are most complementary to the monocular cue, and that user annotation is more useful than selection alone. Figure 4.3 shows example depth reconstructions by our method for several applications.

Table 4.4 and Table 4.5 provide comparisons to a number of other depth completion methods. Two of these do not require task-specific training—Levin *et al.*’s colorization method [88], and Wang *et al.*’s [162] approach to back-propagating errors from measurements. As Wang *et al.*’s own results were with older monocular networks, for a fairer comparison, we derive improved results by applying their method on the same DORN [49] model as used by our network (finding optimal settings on a val set). As seen in Table 4.4 and Table 4.5, our approach is more accurate than both these methods.

Setting	Method	lower is better			higher is better		
		rms	m-rms	rel	$\delta_1$	$\delta_2$	$\delta_3$
Depth Un-cropping (Setting = measurement FOV)							
Horizontal line	Liao [98]	0.442	-	0.104	87.8	96.4	98.9
	Levin [88]	1.003	0.852	0.281	63.8	83.2	92.3
	Wang [162]	0.482	0.394	0.089	90.7	97.3	99.1
	Ours	<b>0.431</b>	<b>0.356</b>	<b>0.088</b>	<b>91.1</b>	<b>98.1</b>	<b>99.5</b>
*120 × 160	Levin [88]	1.104	0.953	0.348	57.5	79.2	90.0
	Wang [162]	0.493	0.409	0.097	89.1	96.9	98.9
	Ours	<b>0.447</b>	<b>0.374</b>	<b>0.097</b>	<b>89.5</b>	<b>97.7</b>	<b>99.3</b>
*240 × 320	Levin [88]	0.664	0.578	0.196	74.2	91.8	96.7
	Wang [162]	0.416	0.342	0.081	91.5	97.7	99.2
	Ours	<b>0.363</b>	<b>0.298</b>	<b>0.076</b>	<b>92.5</b>	<b>98.3</b>	<b>99.5</b>
* Metrics computed only on filled-in regions.							
Depth Up-sampling (Setting = up-sampling factor)							
96x	Chen [29]	0.318	-	0.061	94.2	98.9	<b>99.8</b>
	Levin [88]	0.512	0.443	0.120	85.9	97.1	99.4
	Wang [162]	0.367	0.296	0.057	95.4	98.7	99.6
	Ours	<b>0.313</b>	<b>0.259</b>	<b>0.056</b>	<b>95.7</b>	<b>99.2</b>	<b>99.8</b>
48x	Chen [29]	<b>0.193</b>	-	<b>0.032</b>	<b>98.3</b>	<b>99.7</b>	<b>99.9</b>
	Levin [88]	0.319	0.275	0.065	95.4	99.1	99.8
	Wang [162]	0.318	0.256	0.048	96.7	99.2	99.8
	Ours	0.235	<b>0.195</b>	0.035	97.7	99.6	<b>99.9</b>

Table 4.4: Part A of results for various applications on the NYUv2 test set. We use distributional outputs from our common model to generate depth estimates in a diverse variety of application settings when different forms of additional depth cues are available. We compare to other methods for these applications, including those (shaded background) dependent on task-specific networks trained separately for each setting. Our network, in contrast, is task-agnostic and trained only once.

We also compare to application-specific approaches that train specialized networks separately for each application (and each setting). For depth completion from sparse measurements, we compare to the work of Chen *et al.* [29] for measurements on a regular grid, and of Ma *et al.* [111]<sup>5</sup> for those at random locations. For estimation from horizontal line measurements,

<sup>5</sup>[111] uses a non-standard resolution and crop to evaluate their method and report errors. We report our performance with official settings here to be consistent with the benchmark and the other applications.

Setting	Method	lower is better			higher is better		
		rms	m-rms	rel	$\delta_1$	$\delta_2$	$\delta_3$
Arbitrary Sparse Measurements (Setting = #measurements)							
20	Ma [111]	-	0.351	0.078	92.8	98.4	99.6
	Levin [88]	0.703	0.602	0.175	75.5	93.0	97.9
	Wang [162]	0.399	0.322	<b>0.065</b>	<b>94.2</b>	98.4	99.5
	<i>Ours</i>	<b>0.359</b>	<b>0.298</b>	0.068	94.1	<b>98.8</b>	<b>99.7</b>
50	Ma [111]	-	0.281	0.059	95.5	99.0	99.7
	Levin [88]	0.507	0.436	0.117	86.4	97.1	99.3
	Wang [162]	0.364	0.291	<b>0.056</b>	95.5	98.8	99.6
	<i>Ours</i>	<b>0.320</b>	<b>0.262</b>	<b>0.056</b>	<b>95.6</b>	<b>99.1</b>	<b>99.8</b>
100	Levin [88]	0.396	0.340	0.085	92.2	98.5	99.6
	Wang [162]	0.336	0.271	0.052	96.2	99.0	99.7
	<i>Ours</i>	<b>0.279</b>	<b>0.231</b>	<b>0.046</b>	<b>96.6</b>	<b>99.4</b>	<b>99.9</b>
200	Ma [111]	-	0.230	0.044	97.1	99.4	99.8
	Levin [88]	0.305	0.264	0.061	95.7	99.2	99.8
	Wang [162]	0.316	0.254	0.048	96.6	99.2	99.6
	<i>Ours</i>	<b>0.246</b>	<b>0.203</b>	<b>0.039</b>	<b>97.4</b>	<b>99.5</b>	<b>99.9</b>
User Selection (Setting = #choices)							
5	<i>Ours</i>	0.471	0.406	0.113	87.1	97.4	99.3
10	<i>Ours</i>	0.457	0.394	0.109	87.9	97.6	99.4
15	<i>Ours</i>	0.447	0.385	0.108	88.3	97.8	99.4
User Selection with Annotation (Setting = #choices)							
5	<i>Ours</i>	0.398	0.342	0.098	90.4	98.2	99.6
10	<i>Ours</i>	0.372	0.322	0.093	91.5	98.5	99.7
15	<i>Ours</i>	0.364	0.315	0.090	91.9	98.7	99.7

Table 4.5: Part B of results for various applications on the NYUv2 test set. We use distributional outputs from our common model to generate depth estimates in a diverse variety of application settings when different forms of additional depth cues are available. We compare to other methods for these applications, including those (shaded background) dependent on task-specific networks trained separately for each setting. Our network, in contrast, is task-agnostic and trained only once.

we show comparisons to the method by Liao *et al.* [98]<sup>6</sup>. We find that our results—from a

<sup>6</sup>[98] uses measurements along a line simulated to be horizontal in 3D, leading to different  $y$  image co-ordinates for each  $x$ . Lacking exact details for replicating their setting, we use the same number of measurements but from a line that is horizontal simply in the image plane.

common task-agnostic network model—are comparable, and indeed often better, than these application-specific methods.

#### 4.3.4 Analysis and ablation

We visualize the diversity of depth hypotheses in our distribution in Figure 4.4. We choose one sample for each patch—based on its rank among samples for that patch in terms of accuracy relative to ground-truth. We vary this rank from best to worse, form a global depth map for each rank by overlap-average, and plot the resulting accuracies. Given the ambiguity of the monocular cue, these span a diverse range—from a very accurate estimate when an oracle allows ideal selection, to higher errors when adversarially choosing the worst samples in every patch.

Figure 4.4 also overlays the performance of several of our inference tasks from Table 4.4 and Table 4.5. As expected, the accuracy of pure monocular estimation is roughly at the center of the distribution range. But when additional depth cues are available, we see that our results begin to shift to have higher accuracy—by different amounts for different applications. This shows that our inference method is successful in incorporating the information present in these depth cues.

We also study different variations to our approach for generating samples for our distribution  $p(\mathbf{Z}|\mathbf{I})$  in Table 4.6—measuring performance, on a validation set, in terms of accuracy for a ground truth-based oracle as described above, and more realistically, accuracy at monocular estimation and depth completion (from 100 measurements).

First, we evaluate using a conditional GAN [121] instead of a VAE. While the VAE performs better, results with the GAN are also reasonable—suggesting that our approach is compatible with different network-based sampling approaches.

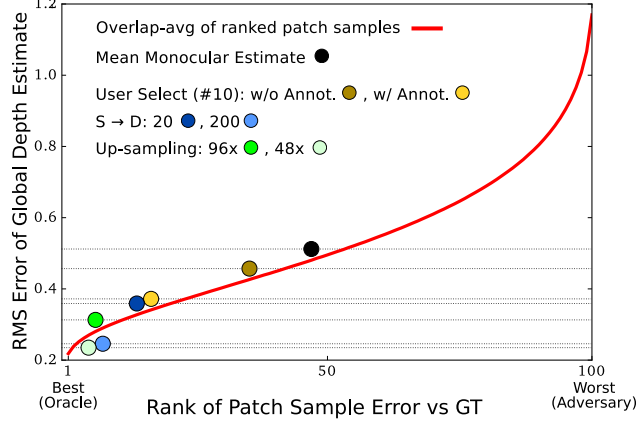


Figure 4.4: Analysis of distributional output and inference method on the test set. Our distribution allows for many possible global depth explanations, visualized here by choosing one of the generated samples in each patch based on the rank of its accuracy going from best (oracle) to worst (adversary), and computing global depth by overlap-average. These solutions span a large range in accuracy, and without any additional information, the mean monocular estimate lies in the middle of this range. But when additional cues are available, they can be effectively exploited by our *MAP* estimation method to extract better solutions from our distribution.

Then, we consider varying the size of our patches (and proportionally, the stride). We find smaller patches actually helps oracle performance, since with the same number of samples, it is easier to generate a sample close to the ground-truth in a lower-dimensional space. However, smaller patches do not accurately capture the spatial dependencies within a patch, leading to poorer performance for actual inference. Conversely, while a higher patch size could allow encoding longer range spatial dependencies, doing so is harder via approximation from a reasonable number of samples—leading to lower accuracy both with the oracle and during inference.

For our chosen patch-size, we also evaluate higher strides, and thus lower overlap. This leads to lower performance (on depth completion), highlighting the utility of patch-overlap in the global distribution  $p(\mathbf{Z}|\mathbf{I})$ , and in propagating information during inference.



		Oracle	Mean	S→D		
C-GAN	p=33,s=4	0.384	0.597	0.428	C-VAE p=33	S→D
C-VAE	p=17,s=2	<b>0.263</b>	0.518	0.413	s=8	0.396
<b>C-VAE</b>	<b>p=33,s=4</b>	0.323	<b>0.516</b>	<b>0.377</b>	s=16	0.405
C-VAE	p=65,s=8	0.474	0.522	0.389	s=32	0.436

Table 4.6: Ablation study on validation set. We evaluate different ways of generating samples: using a GAN instead of a VAE, and using different patch-sizes  $p$  (with proportional strides  $s$ ). For each case, we compare achievable accuracy of individual samples via the “oracle” estimate (see Figure 4.4), vs. their utility for actual inference—in the pure monocular case and with random sparse measurements ( $\#100$ ). We also evaluate the importance of patch overlap by considering larger strides for our chosen model.

## 4.4 Discussion

With distributional monocular outputs, our approach enables a variety of applications without the need for repeated training. While we considered tasks directly focused on scene geometry in this work, It would also be interesting to explore how our distributional outputs can be used to manage ambiguity in downstream processing—such as for re-rendering or path planning—in future work. We also believe probabilistic predictions can be useful for other low- and mid-level scene properties, like motion and reflectance.

# Chapter 5

## Conclusion

The prevalence of mobile photography and the culture of content sharing has incited dramatic changes in the way we perceive the world. Our stares are focused on screens, more often than ever, to see the world that is far away physically and temporally via captured images and videos. While this allows us to personally experience sights from around the world, it also opens up opportunities to use this information computationally to enable a whole new class of applications. The motivation for this research has been on advancing algorithms to address some of the challenges in this context.

In this dissertation, we discussed how to reconstruct a variety of physical and visual properties of the world from these captured images, especially in the context of deep learning. We exploited our prior understanding on the internal structure of natural scene maps and physics-based image models to design and train neural network based approaches for scene map estimations. We proposed novel methods based on the internal structure of natural images to recover clean images in low-light environments. We also tackled the notorious problem of data insufficiency for computational photography applications by training scene estimators with

indirect scene measurements. Finally, we recognized that for all of these applications, our estimations will never be perfect and there are other sources for scene information available in practice. We demonstrated that we can derive a rich representation of our understanding of the scene parameters given observed images, which can be combined with additional information.

Over the last decade, we have witnessed rapid and exciting advances in computer vision and computational photography research. While data is powerful, we have seen that using generic neural networks architectures as black boxes and relying on data alone is insufficient, especially for the applications we considered in this dissertation. However, the problem of decoding scene properties from visual measurements are far from settled. While the techniques described here shed some light on how to address the challenges of scene map estimation, they are only the beginnings of many possible interesting directions of exploration.

Importantly, many more and new challenges have emerged as we are getting closer to our goal to reconstruct all dimensions of the scene from images taken with cameras by casual users. Computational photography applications are no longer satisfied with high-quality images, their results must be photo-realistic for users to capture professional-level photos with a pocket phone. Even more ambitiously, mobile photography techniques are seeking to automatically tune the camera setting and even virtually “edit” the scene to produce better images, such as viewfinder with auto exposure [57], shadow removal [188] and light editing [11]. Meanwhile, the path of making machine learning models for computational photography faster and more efficient has just begun. It is critical to enable these processing algorithms to run on small devices with limited power and computational resources, such as mobile phones and AR glasses.

Numerous effort, on the other hand, are invested in solving the insufficiency of training data for scene map estimations both in the academia and by major tech companies. Facebook has recently launched the so-called project Aria [42] where people wear glasses that are equipped with cameras and sensors and walk around to capture images while record the world. Contrast with the lack of training data is the massive amount of images and scene measurements that are produced by regular users. For example, over 1.4 trillion photos were taken in the year of 2020 alone [129]. Self-driving cars with many sensors are deployed on the road for data acquisition [165]. Gaming stations with depth sensors are collecting depth maps for gaming experiences [152]. How to make use of such unpaired data and imperfect measurements of the world to solve each scene map estimation application remains an open research question.

Last but not least, reliable machine learning models are desired for their robustness and consistency. Scene map estimators must be able to faithfully describe its reasoning and ambiguity of the scene, and resolve them by the integration into a larger system where acquisitions devices other than cameras are available. The modularization of scene map estimators could open broader opportunities in a wide variety of downstream applications.

In the coming years, we will likely see machine intelligence, together with our understanding of the visual world, revolutionize the way people produce and exploit image data, and eventually how we experience the world.

# References

- [1] Edward H Adelson and Alex P Pentland. “The perception of shading and reflectance”. In: *Perception as Bayesian inference* (1996), pp. 409–423.
- [2] Yagiz Aksoy, Changil Kim, Petr Kellnhofer, Sylvain Paris, Mohamed A. Elgharib, Marc Pollefeys, and Wojciech Matusik. “A Dataset of Flash and Ambient Illumination Pairs from the Crowd”. In: *Proc. ECCV*. 2018.
- [3] Rushil Anirudh, Jayaraman J Thiagarajan, Bhavya Kailkhura, and Timo Bremer. “An unsupervised approach to solving inverse problems using generative adversarial networks”. In: *arXiv preprint arXiv:1805.07281* (2018).
- [4] Jonathan T. Barron and Jitendra Malik. “Intrinsic Scene Properties from a Single RGB-D Image”. In: *Proc. CVPR* (2013).
- [5] Jonathan T Barron and Jitendra Malik. “Shape, albedo, and illumination from a single image of an unknown object”. In: *Proc. CVPR*. 2012.
- [6] Jonathan T. Barron and Jitendra Malik. “Shape, Illumination, and Reflectance from Shading”. In: *TPAMI* (2015).
- [7] H. Barrow and J. M. Tenenbaum. “RECOVERING INTRINSIC SCENE CHARACTERISTICS FROM IMAGES”. In: 1978.
- [8] Anil S. Baslamisli, Hoang-An Le, and Theo Gevers. “CNN Based Learning Using Reflection and Retinex Models for Intrinsic Image Decomposition”. In: *Proc. CVPR*. 2018.
- [9] Dhruv Batra, Payman Yadollahpour, Abner Guzman-Rivera, and Gregory Shakhnarovich. “Diverse M-Best solutions in Markov Random Fields”. In: *Proc. ECCV*. 2012.
- [10] Sean Bell, Kavita Bala, and Noah Snavely. “Intrinsic Images in the Wild”. In: *ACM Transactions on Graphics (TOG)* 33.4 (2014).
- [11] Sai Bi, Zexiang Xu, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. “Deep reflectance volumes: Relightable reconstructions from multi-view photometric images”. In: *Proc. ECCV*. 2020.

- [12] Volker Blanz and Thomas Vetter. “A morphable model for the synthesis of 3D faces”. In: *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. 1999.
- [13] James F. Blinn. “Models of Light Reflection for Computer Synthesized Pictures”. In: *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques*. 1977, pp. 192–198.
- [14] Ashish Bora, Eric Price, and Alexandros G Dimakis. “AmbientGAN: Generative models from lossy measurements”. In: *Proc. ICLR*. 2018.
- [15] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. “A Non-Local Algorithm for Image Denoising”. In: *Proc. CVPR*. 2005.
- [16] Antoni Buades, Bartomeu Coll, and J-M Morel. “A non-local algorithm for image denoising”. In: *Proc. CVPR*. 2005.
- [17] Antoni Buades, Bartomeu Coll, and J-M Morel. “A non-local algorithm for image denoising”. In: *Proc. CVPR*. 2005.
- [18] Harold C Burger, Christian J Schuler, and Stefan Harmeling. “Image denoising: Can plain neural networks compete with BM3D?” In: *Proc. CVPR*. 2012.
- [19] Xu Cao, Michael Waechter, Boxin Shi, Ye Gao, Bo Zheng, and Yasuyuki Matsushita. “Stereoscopic Flash and No-Flash Photography for Shape and Albedo Recovery”. In: *Proc. CVPR*. 2020.
- [20] Ayan Chakrabarti. “A neural approach to blind motion deblurring”. In: *Proc. ECCV*. 2016.
- [21] Ayan Chakrabarti, Jingyu Shao, and Greg Shakhnarovich. “Depth from a single image by harmonizing overcomplete local network predictions”. In: *NeurIPS*. 2016.
- [22] Jen-Hao Rick Chang, Chun-Liang Li, Barnabas Poczos, BVK Vijaya Kumar, and Aswin C Sankaranarayanan. “One Network to Solve Them All-Solving Linear Inverse Problems using Deep Projection Models.” In: *Proc. ICCV*. 2017.
- [23] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. “Learning to See in the Dark”. In: *Proc. CVPR*. 2018.
- [24] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. “Rethinking Atrous Convolution for Semantic Image Segmentation”. In: *CoRR* abs/1706.05587 (2017).
- [25] Qifeng Chen, Jia Xu, and Vladlen Koltun. “Fast Image Processing with Fully-Convolutional Networks”. In: *Proc. ICCV*. 2017.
- [26] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. “Single-image depth perception in the wild”. In: *NeurIPS*. 2016.
- [27] Yunjin Chen and Thomas Pock. “Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration”. In: *TPAMI* (2017).

- [28] Yunjin Chen and Thomas Pock. “Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration”. In: *TPAMI* 39.6 (2017), pp. 1256–1272.
- [29] Zhao Chen, Vijay Badrinarayanan, Gilad Drozdov, and Andrew Rabinovich. “Estimating depth from RGB and sparse sensing”. In: *Proc. ECCV*. 2018.
- [30] Ziang Cheng, Yinqiang Zheng, Shaodi You, and Imari Sato. “Non-local intrinsic decomposition with near-infrared priors”. In: *Proc. ICCV*. 2019.
- [31] Gyeongmin Choe, Jaesik Park, Yu-Wing Tai, and In Kweon. “Exploiting shading cues in kinect ir images for geometry refinement”. In: *Proc. CVPR*. 2014.
- [32] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. “Color image denoising via sparse 3D collaborative filtering with grouping constraint in luminance-chrominance space”. In: *Proc. ICIP*. 2007.
- [33] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. “Image denoising by sparse 3-D transform-domain collaborative filtering”. In: *IEEE Transactions on Image Processing* (2007).
- [34] Yu Deng, Jiaolong Yang, Sicheng Xu, Dong Chen, Yunde Jia, and Xin Tong. “Accurate 3d face reconstruction with weakly-supervised learning: From single image to image set”. In: *Proc. CVPR Workshops*. 2019.
- [35] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. “Image super-resolution using deep convolutional networks”. In: *TPAMI* 38.2 (2015), pp. 295–307.
- [36] David L Donoho et al. “Compressed sensing”. In: *IEEE Transactions on Information Theory* 52.4 (2006), pp. 1289–1306.
- [37] David L. Donoho. “De-noising by soft-thresholding”. In: *IEEE Transactions on Information Theory* 41.3 (1995), pp. 613–627.
- [38] David Doria and Richard J Radke. “Filling large holes in lidar data by inpainting depth gradients”. In: *Proc. CVPR Workshops*. 2012.
- [39] David Eigen and Rob Fergus. “Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture”. In: *Proc. ICCV*. 2015.
- [40] David Eigen, Christian Puhrsch, and Rob Fergus. “Depth map prediction from a single image using a multi-scale deep network”. In: *NeurIPS*. 2014.
- [41] Elmar Eisemann and Frédo Durand. “Flash photography enhancement via intrinsic relighting”. In: *ACM Transactions on Graphics (TOG)* 23.3 (2004), pp. 673–678.
- [42] *Facebook Project Aria*. <https://about.fb.com/realitylabs/projectaria/>.
- [43] Qingnan Fan, Jiaolong Yang, Gang Hua, Baoquan Chen, and David Wipf. “Revisiting deep intrinsic image decompositions”. In: *Proc. CVPR*. 2018, pp. 8944–8952.
- [44] Mário AT Figueiredo, Robert D Nowak, and Stephen J Wright. “Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems”. In: *IEEE Journal of selected topics in signal processing* 1.4 (2007), pp. 586–597.

- [45] Chelsea Finn, Ian Goodfellow, and Sergey Levine. “Unsupervised learning for physical interaction through video prediction”. In: *Advances in neural information processing systems*. 2016, pp. 64–72.
- [46] Alessandro Foi, Mejdi Trimeche, Vladimir Katkovnik, and Karen Egiazarian. “Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data”. In: *IEEE Transactions on Image Processing* 17.10 (2008), pp. 1737–1754.
- [47] Rich Franzen. “Kodak lossless true color image suite”. In: *source: <http://r0k.us/graphics/kodak>* 4 (1999).
- [48] William T Freeman, Thouis R Jones, and Egon C Pasztor. “Example-based super-resolution”. In: *IEEE Computer Graphics and Applications* 2 (2002), pp. 56–65.
- [49] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. “Deep ordinal regression network for monocular depth estimation”. In: *Proc. CVPR*. 2018, pp. 2002–2011.
- [50] Ravi Garg, Vijay Kumar BG, Gustavo Carneiro, and Ian Reid. “Unsupervised CNN for single view depth estimation: Geometry to the rescue”. In: *Proc. ECCV*. 2016.
- [51] Michaël Gharbi, Jiawen Chen, Jonathan T. Barron, Samuel W. Hasinoff, and Frédo Durand. “Deep bilateral learning for real-time image enhancement”. In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), 118:1–118:12.
- [52] Clément Godard, Oisín Mac Aodha, and Gabriel J Brostow. “Unsupervised monocular depth estimation with left-right consistency”. In: *Proc. CVPR*. 2017.
- [53] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. “Generative adversarial nets”. In: *NeurIPS*. 2014.
- [54] Shuhang Gu, Wangmeng Zuo, Shi Guo, Yunjin Chen, Chongyu Chen, and Lei Zhang. “Learning dynamic guidance for depth image enhancement”. In: *Proc. CVPR*. 2017.
- [55] Prabath Gunawardane, Tom Malzbender, Ramin Samadani, Alan McReynolds, Dan Gelb, and James Davis. “Invisible light: Using infrared for video conference relighting”. In: *Proc. ICIP*. 2010.
- [56] Qiang Guo, Caiming Zhang, Yunfeng Zhang, and Hui Liu. “An Efficient SVD-Based Method for Image Denoising”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 26.5 (2016), pp. 868–880.
- [57] Samuel W Hasinoff, Dillon Sharlet, Ryan Geiss, Andrew Adams, Jonathan T Barron, Florian Kainz, Jiawen Chen, and Marc Levoy. “Burst photography for high dynamic range and low-light imaging on mobile cameras”. In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), p. 192.
- [58] Felix Heide, Steven Diamond, Matthias Nießner, Jonathan Ragan-Kelley, Wolfgang Heidrich, and Gordon Wetzstein. “Proximal: Efficient image optimization using proximal algorithms”. In: *ACM Transactions on Graphics (TOG)* 35.4 (2016), p. 84.



- [59] Felix Heide et al. “FlexISP: A flexible camera image processing framework”. In: *ACM Transactions on Graphics (TOG)* 33.6 (2014), p. 231.
- [60] Minhyeok Heo, Jaehan Lee, Kyung-Rae Kim, Han-Ul Kim, and Chang-Su Kim. “Monocular Depth Estimation Using Whole Strip Masking and Reliability-Based Refinement”. In: *Proc. ECCV*. 2018.
- [61] Daniel Herrera, Juho Kannala, Janne Heikkilä, et al. “Depth map inpainting under a second-order smoothness prior”. In: *Scandinavian Conference on Image Analysis*. 2013.
- [62] Berthold K. P. Horn. “Obtaining Shape from Shading Information”. In: *Shape from Shading*. Cambridge, MA, USA: MIT Press, 1989, pp. 123–171.
- [63] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. “Densely connected convolutional networks”. In: *Proc. CVPR*. 2017.
- [64] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. “Single image super-resolution from transformed self-exemplars”. In: *Proc. CVPR*. 2015.
- [65] Maximilian Jaritz, Raoul De Charette, Emilie Wirbel, Xavier Perrotton, and Fawzi Nashashibi. “Sparse and dense data with CNNs: Depth completion and semantic segmentation”. In: *Proc. 3DV*. 2018.
- [66] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. “Dynamic filter networks”. In: *NeurIPS*. 2016.
- [67] Younghyun Jo, Seoung Wug Oh, Jaeyeon Kang, and Seon Joo Kim. “Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation”. In: *Proc. CVPR*. 2018, pp. 3224–3232.
- [68] Kevin Karsch, Ce Liu, and Sing Bing Kang. “Depth transfer: Depth extraction from video using non-parametric sampling”. In: *TPAMI* (2014).
- [69] Alex Kendall and Yarin Gal. “What uncertainties do we need in bayesian deep learning for computer vision?” In: *NeurIPS*. 2017.
- [70] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *Proc. ICLR*. 2015.
- [71] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
- [72] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
- [73] Simon Kohl et al. “A probabilistic U-Net for segmentation of ambiguous images”. In: *NeurIPS*. 2018.
- [74] Filippos Kokkinos and Stamatis Lefkimmiatis. “Iterative residual cnns for burst photography applications”. In: *Proc. CVPR*. 2019, pp. 5929–5938.

- [75] Janusz Konrad, Meng Wang, Prakash Ishwar, Chen Wu, and Debargha Mukherjee. “Learning-based, automatic 2D-to-3D image and video conversion”. In: *IEEE Trans. on Image Processing* (2013).
- [76] Ivan Krasin et al. “OpenImages: A public dataset for large-scale multi-label and multi-class image classification.” In: *Dataset available from <https://github.com/openimages>* (2017).
- [77] D. Krishnan and R. Fergus. “Dark Flash Photography”. In: *ACM Transactions on Graphics (TOG)* 28.3 (2009).
- [78] Dilip Krishnan and Rob Fergus. “Dark flash photography”. In: *ACM Transactions on Graphics (TOG)* 28.3 (2009), p. 96.
- [79] Kuldeep Kulkarni, Suhas Lohit, Pavan Turaga, Ronan Kerviche, and Amit Ashok. “Reconnet: Non-iterative reconstruction of images from compressively sensed measurements”. In: *Proc. CVPR*. 2016.
- [80] Yevhen Kuznetsov, Jorg Stuckler, and Bastian Leibe. “Semi-supervised deep learning for monocular depth map prediction”. In: *Proc. CVPR*. 2017.
- [81] Lubor Ladicky, Jianbo Shi, and Marc Pollefeys. “Pulling things out of perspective”. In: *Proc. CVPR*. 2014.
- [82] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. “Deeper depth prediction with fully convolutional residual networks”. In: *Proc. 3DV*. 2016.
- [83] Vuong Ba Lê, Jonathan Brandt, Zhe L. Lin, Lubomir D. Bourdev, and Thomas S. Huang. “Interactive Facial Feature Localization”. In: *Proc. ECCV*. 2012.
- [84] Jae-Han Lee, Minhyeok Heo, Kyung-Rae Kim, and Chang-Su Kim. “Single-image depth estimation based on Fourier domain analysis”. In: *Proc. CVPR*. 2018.
- [85] Jae-Han Lee and Chang-Su Kim. “Monocular Depth Estimation Using Relative Depth Maps”. In: *Proc. CVPR*. 2019.
- [86] Stamatios Lefkimmiatis. “Non-local color image denoising with convolutional neural networks”. In: *Proc. CVPR* (2017).
- [87] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. “Noise2noise: Learning image restoration without clean data”. In: *arXiv preprint [arXiv:1803.04189](https://arxiv.org/abs/1803.04189)* (2018).
- [88] Anat Levin, Dani Lischinski, and Yair Weiss. “Colorization using optimization”. In: *ACM Transactions on Graphics (TOG)*. 2004.
- [89] Chengbo Li, Wotao Yin, Hong Jiang, and Yin Zhang. “An efficient augmented Lagrangian method with applications to total variation minimization”. In: *Computational Optimization and Applications* 56.3 (2013), pp. 507–530.

- [90] Huibin Li and Feng Liu. “Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries in Wavelet Domain”. In: *Proc. International Conference on Image and Graphics (ICIG)*. 2009.
- [91] Jun Li, Reinhard Klein, and Angela Yao. “A Two-Streamed Network for Estimating Fine-Scaled Depth Maps From Single RGB Images”. In: *Proc. ICCV*. 2017.
- [92] Yijun Li, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. “Deep Joint Image Filtering”. In: *Proc. ECCV*. 2016.
- [93] Yijun Li, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. “Deep joint image filtering”. In: *Proc. ECCV*. 2016.
- [94] Zhengqi Li and Noah Snavely. “Learning intrinsic image decomposition from watching the world”. In: *Proc. CVPR*. 2018.
- [95] Zhengqin Li, Mohammad Shafiei, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. “Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image”. In: *Proc. CVPR*. 2020, pp. 2475–2484.
- [96] Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. “Learning to Reconstruct Shape and Spatially-Varying Reflectance from a Single Image”. In: *ACM Transactions on Graphics (TOG)* 37.6 (2018).
- [97] Zhe Liang, Chao Xu, Jing Hu, Yushi Li, and Zhaopeng Meng. “Better Together: Shading Cues and Multi-View Stereo for Reconstruction Depth Optimization”. In: *IEEE Access* PP (2020), pp. 1–1. DOI: 10.1109/ACCESS.2020.3003023.
- [98] Yiyi Liao, Lichao Huang, Yue Wang, Sarath Kodagoda, Yinan Yu, and Yong Liu. “Parse geometry from a line: Monocular depth estimation with partial laser observation”. In: *Proc. ICRA*. 2017.
- [99] Orly Liba et al. “Handheld Mobile Photography in Very Low Light”. In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), 164:1–164:16.
- [100] Michael Lindenbaum, M. Fischer, and Alfred M. Bruckstein. “On Gabor’s contribution to image enhancement”. In: *Pattern Recognition* 27.1 (1994), pp. 1–8.
- [101] Chao Liu, Jinwei Gu, Kihwan Kim, Srinivasa Narasimhan, and Jan Kautz. “Neural RGB->D Sensing: Depth and Uncertainty from a Video Camera”. In: *arXiv preprint arXiv:1901.02571* (2019).
- [102] Ding Liu, Bihan Wen, Yuchen Fan, Chen Change Loy, and Thomas S Huang. “Non-Local Recurrent Network for Image Restoration”. In: *NeurIPS*. 2018.
- [103] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. “Learning depth from single monocular images using deep convolutional neural fields”. In: *TPAMI* (2016).
- [104] Junyi Liu and Xiaojin Gong. “Guided depth enhancement via anisotropic diffusion”. In: *Pacific-Rim Conference on Multimedia*. 2013.
- [105] Junyi Liu, Xiaojin Gong, and Jilin Liu. “Guided inpainting and filtering for Kinect depth maps”. In: *Proc ICPR*. 2012.

- [106] Miaomiao Liu, Mathieu Salzmann, and Xuming He. “Discrete-continuous depth estimation from a single image”. In: *Proc. CVPR*. 2014.
- [107] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. “Deep Learning Face Attributes in the Wild”. In: *Proc. ICCV*. 2015.
- [108] Ziwei Liu, Raymond A Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. “Video frame synthesis using deep voxel flow”. In: *Proc. ICCV*. 2017, pp. 4463–4471.
- [109] Ziwei Liu, Lu Yuan, Xiaoou Tang, Matt Uyttendaele, and Jian Sun. “Fast burst images denoising”. In: *ACM Transactions on Graphics (TOG)* 33.6 (2014), p. 232.
- [110] Michael Lustig, David L Donoho, Juan M Santos, and John M Pauly. “Compressed sensing MRI”. In: *IEEE Signal Processing* 25.2 (2008), p. 72.
- [111] Fangchang Ma and Sertac Karaman. “Sparse-to-dense: Depth prediction from sparse depth samples and a single image”. In: *Proc. ICRA*. 2018.
- [112] Kede Ma, Zhengfang Duanmu, Qingbo Wu, Zhou Wang, Hongwei Yong, Hongliang Li, and Lei Zhang. “Waterloo exploration database: New challenges for image quality assessment models”. In: *IEEE Transactions on Image Processing* (2017).
- [113] Wei-Chiu Ma, Hang Chu, Bolei Zhou, Raquel Urtasun, and Antonio Torralba. “Single image intrinsic decomposition without a single intrinsic image”. In: *Proc. ECCV*. 2018.
- [114] Matteo Maggioni, Giacomo Boracchi, Alessandro Foi, and Karen Egiazarian. “Video denoising, deblocking, and enhancement through separable 4-D nonlocal spatiotemporal transforms”. In: *IEEE Transactions on Image Processing* 21.9 (2012), pp. 3952–3966.
- [115] Talmaj Marinč, Vignesh Srinivasan, Serhan Gül, Cornelius Hellge, and Wojciech Samek. “Multi-Kernel Prediction Networks for Denoising of Burst Images”. In: *arXiv preprint arXiv:1902.05392* (2019).
- [116] D. Martin, C. Fowlkes, D. Tal, and J. Malik. “A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics”. In: *Proc. ICCV*. 2001.
- [117] Kiyoshi Matsuo and Yoshimitsu Aoki. “Depth Image Enhancement Using Local Tangent Plane Approximations”. In: *Proc. CVPR*. 2015.
- [118] Christopher A Metzler, Arian Maleki, and Richard G Baraniuk. “From denoising to compressed sensing”. In: *IEEE Transactions on Information Theory* 62.9 (2016), pp. 5117–5144.
- [119] Christopher A Metzler, Ali Mousavi, Reinhard Heckel, and Richard G Baraniuk. “Unsupervised Learning with Stein’s Unbiased Risk Estimator”. In: *arXiv preprint arXiv:1805.10531* (2018).
- [120] Ben Mildenhall, Jonathan T Barron, Jiawen Chen, Dillon Sharlet, Ren Ng, and Robert Carroll. “Burst denoising with kernel prediction networks”. In: *Proc. CVPR*. 2018, pp. 2502–2510.

- [121] Mehdi Mirza and Simon Osindero. “Conditional generative adversarial nets”. In: *arXiv preprint arXiv:1411.1784* (2014).
- [122] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. “Deep Multi-scale Convolutional Neural Network for Dynamic Scene Deblurring”. In: *Proc. CVPR* (2017).
- [123] Diego Nehab, Szymon Rusinkiewicz, James Davis, and Ravi Ramamoorthi. “Efficiently combining positions and normals for precise 3D geometry”. In: *ACM transactions on graphics (TOG)* 24.3 (2005), pp. 536–543.
- [124] Thomas Nestmeyer, Jean-François Lalonde, Iain Matthews, and Andreas M Lehrmann. “Learning Physics-guided Face Relighting under Directional Light”. In: *Proc. CVPR*. 2020.
- [125] Justin Ng. *Yes, The Huawei P30 Pro Can Shoot the Milky Way*. <https://petapixel.com/2019/05/13/shooting-the-milky-way-and-meteors-with-the-huawei-p30-pro/>.
- [126] Simon Niklaus, Long Mai, and Feng Liu. “Video frame interpolation via adaptive convolution”. In: *Proc. CVPR*. 2017, pp. 670–679.
- [127] Simon Niklaus, Long Mai, and Feng Liu. “Video frame interpolation via adaptive separable convolution”. In: *Proc. ICCV*. 2017, pp. 261–270.
- [128] Harris Nover, Supreeth Achar, and Dan B Goldman. “ESPreSSo: Efficient Slanted PatchMatch for Real-time Spacetime Stereo”. In: *Proc. 3DV*. 2018.
- [129] *Number of photos taken per year*. <https://focus.mylio.com/tech-today/how-many-photos-will-be-taken-in-2021>.
- [130] Pietro Perona and Jitendra Malik. “Scale-Space and Edge Detection Using Anisotropic Diffusion”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12.7 (1990), pp. 629–639.
- [131] Georg Petschnigg, Richard Szeliski, Maneesh Agrawala, Michael Cohen, Hugues Hoppe, and Kentaro Toyama. “Digital Photography with Flash and No-Flash Image Pairs”. In: *ACM Transactions on Graphics (TOG)* 23.3 (2004), pp. 664–672.
- [132] Di Qiu, Jin Zeng, Zhanghan Ke, Wenxiu Sun, and Chengxi Yang. “Towards Geometry Guided Neural Relighting with Flash Photography”. In: *arXiv preprint arXiv:2008.05157* (2020).
- [133] Rene Ranftl, Vibhav Vineet, Qifeng Chen, and Vladlen Koltun. “Dense monocular depth estimation in complex dynamic scenes”. In: *Proc. CVPR*. 2016.
- [134] Yaniv Romano, Michael Elad, and Peyman Milanfar. “The little engine that could: Regularization by denoising (RED)”. In: *SIAM Journal on Imaging Sciences* 10.4 (2017), pp. 1804–1844.
- [135] O. Ronneberger, P. Fischer, and T. Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Vol. 9351. 2015, pp. 234–241.

- [136] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. 2015, pp. 234–241.
- [137] Stefan Roth and Michael J Black. “Fields of experts”. In: *IJCV* (2009).
- [138] Anirban Roy and Sinisa Todorovic. “Monocular depth estimation using neural regression forest”. In: *Proc. CVPR*. 2016.
- [139] Leonid I Rudin, Stanley Osher, and Emad Fatemi. “Nonlinear total variation based noise removal algorithms”. In: *Physica D: nonlinear phenomena* 60.1-4 (1992), pp. 259–268.
- [140] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *IJCV* (2015).
- [141] Soubhik Sanyal, Timo Bolkart, Haiwen Feng, and Michael Black. “Learning to Regress 3D Face Shape and Expression From an Image Without 3D Supervision”. In: *Proc. CVPR*. 2019.
- [142] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. “Learning depth from single monocular images”. In: *NeurIPS*. 2006.
- [143] Ashutosh Saxena, Min Sun, and Andrew Y Ng. “Make3d: Learning 3d scene structure from a single still image”. In: *TPAMI* (2009).
- [144] *See the light with Night Sight*. <https://www.blog.google/products/pixel/see-light-night-sight/>.
- [145] Soumyadip Sengupta, Angjoo Kanazawa, Carlos D. Castillo, and David W. Jacobs. “SfSNet: Learning Shape, Reflectance and Illuminance of Faces in the Wild”. In: *Proc. CVPR*. 2018.
- [146] Ziyi Shen, Wei-Sheng Lai, Tingfa Xu, Jan Kautz, and Ming-Hsuan Yang. “Deep semantic face deblurring”. In: *Proc. CVPR*. 2018.
- [147] Jian Shi, Yue Dong, Hao Su, and Stella X. Yu. “Learning Non-Lambertian Object Intrinsic across ShapeNet Categories.” In: *CoRR* abs/1612.08510 (2016).
- [148] Jianping Shi, Xin Tao, Li Xu, and Jiaya Jia. “Break ames room illusion: depth from general single images”. In: *ACM Transactions on Graphics (TOG)* (2015).
- [149] Shreyas S Shivakumar, Ty Nguyen, Steven W Chen, and Camillo J Taylor. “DFuseNet: Deep Fusion of RGB and Sparse Depth Information for Image Guided Dense Depth Completion”. In: *arXiv preprint arXiv:1902.00761* (2019).
- [150] Z. Shu, E. Yumer, S. Hadap, K. Sunkavalli, E. Shechtman, and D. Samaras. “Neural Face Editing with Intrinsic Image Disentangling”. In: *Proc. CVPR*. 2017.
- [151] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. “Indoor segmentation and support inference from rgb-d images”. In: *Proc. ECCV*. 2012.
- [152] *Stereo Camera on PlayStation 5*. <https://www.roadtovr.com/playstation-5-stereo-hd-camera-psvr-2-tracking/>.

- [153] Tiancheng Sun et al. “Single Image Portrait Relighting”. In: *ACM Transactions on Graphics (TOG)* 38.4 (2019).
- [154] Tatsunori Taniai and Takanori Maehara. “Neural Inverse Rendering for General Reflectance Photometric Stereo”. In: *Proc. ICML*. 2018.
- [155] Ayush Tewari, Michael Zollöfer, Hyeonwoo Kim, Pablo Garrido, Florian Bernard, Patrick Perez, and Theobalt Christian. “MoFA: Model-based Deep Convolutional Face Autoencoder for Unsupervised Monocular Reconstruction”. In: *Proc. ICCV*. 2017.
- [156] Carlo Tomasi and Roberto Manduchi. “Bilateral Filtering for Gray and Color Images”. In: *Proc. ICCV*. 1998.
- [157] Anh Tran, Tal Hassner, Iacopo Masi, Eran Paz, Yuval Nirkin, and Gerard Medioni. “Extreme 3D Face Reconstruction: Seeing Through Occlusions”. In: *Proc. CVPR*. 2018.
- [158] Wouter Van Gansbeke, Davy Neven, Bert De Brabandere, and Luc Van Gool. “Sparse and noisy LiDAR completion with RGB guidance and uncertainty”. In: *arXiv preprint arXiv:1902.05356* (2019).
- [159] Jian Wang, Tianfan Xue, Jonathan T. Barron, and Jiawen Chen. “Stereoscopic Dark Flash for Low-light Photography”. In: *Proc. ICCP*. 2019.
- [160] Oliver Wang, James Davis, Erika Chuang, Ian Rickard, Krystle De Mesa, and Chirag Dave. “Video relighting using infrared illumination”. In: *Computer Graphics Forum*. Vol. 27. 2. 2008, pp. 271–279.
- [161] Peng Wang, Xiaohui Shen, Zhe Lin, Scott Cohen, Brian Price, and Alan L Yuille. “Towards unified depth and semantic prediction from a single image”. In: *Proc. CVPR*. 2015.
- [162] Tsun-Hsuan Wang, Fu-En Wang, Juan-Ting Lin, Yi-Hsuan Tsai, Wei-Chen Chiu, and Min Sun. “Plug-and-Play: Improve Depth Prediction via Sparse Data Propagation”. In: *Proc. ICRA*. 2019.
- [163] Xiaolong Wang, David Fouhey, and Abhinav Gupta. “Designing deep networks for surface normal estimation”. In: *Proc. CVPR*. 2015.
- [164] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Transactions on Image Processing* (2004).
- [165] *Waymo Open Dataset*. <https://waymo.com/open/about/>.
- [166] Tim Weyrich et al. “Analysis of Human Faces using a Measurement-Based Skin Reflectance Model”. In: *ACM Transactions on Graphics (TOG)* 25.3 (2006), pp. 1013–1024.
- [167] Bartłomiej Wronski, Ignacio Garcia-Dorado, Manfred Ernst, Damien Kelly, Michael Krainin, Chia-Kai Liang, Marc Levoy, and Peyman Milanfar. “Handheld multi-frame super-resolution”. In: *ACM Transactions on Graphics (TOG)* 38.4 (2019), pp. 1–18.

- [168] Zhihao Xia and Ayan Chakrabarti. “Identifying recurring patterns with deep neural networks for natural image denoising”. In: *arXiv preprint arXiv:1806.05229* (2018).
- [169] Zhihao Xia, Federico Perazzi, Michaël Gharbi, Kalyan Sunkavalli, and Ayan Chakrabarti. “Basis Prediction Networks for Effective Burst Denoising With Large Kernels”. In: *Proc. CVPR*. 2020.
- [170] Ke Xian, Chunhua Shen, Zhiguo Cao, Hao Lu, Yang Xiao, Ruibo Li, and Zhenbo Luo. “Monocular relative depth perception with web stereo data supervision”. In: *Proc. CVPR*. 2018.
- [171] Jiangjian Xiao, Hui Cheng, Harpreet Sawhney, Cen Rao, and Michael Isnardi. “Bilateral Filtering-Based Optical Flow Estimation with Occlusion Detection”. In: *Proc. ECCV*. 2006.
- [172] Junyuan Xie, Linli Xu, and Enhong Chen. “Image Denoising and Inpainting with Deep Neural Networks”. In: *NeurIPS*. 2012.
- [173] Li Xu, Shicheng Zheng, and Jiaya Jia. “Unnatural l0 sparse representation for natural image deblurring”. In: *Proc. CVPR*. 2013.
- [174] H. Yamashita, D. Sugimura, and T. Hamamoto. “RGB-NIR imaging with exposure bracketing for joint denoising and deblurring of low-light color images”. In: *Proc. ICASSP*. 2017.
- [175] Qiong Yan, Xiaoyong Shen, Li Xu, Shaojie Zhuo, Xiaopeng Zhang, Liang Shen, and Jiaya Jia. “Cross-Field Joint Image Restoration via Scale Map”. In: *Proc. ICCV*. 2013.
- [176] Dong Yang and Jian Sun. “BM3D-Net: A Convolutional Neural Network for Transform-Domain Collaborative Filtering”. In: *IEEE Signal Processing Letters* (2018).
- [177] Zhenheng Yang, Peng Wang, Wei Xu, Liang Zhao, and Ramakant Nevatia. “Unsupervised learning of geometry with edge-aware depth-normal consistency”. In: *arXiv preprint arXiv:1711.03665* (2017).
- [178] Leonid P Yaroslavsky. “Digital picture processing: an introduction”. In: *Applied Optics* 25.18 (1986), p. 3127.
- [179] Youngjin Yoon, Gyeongmin Choe, Namil Kim, Joon-Young Lee, and In So Kweon. “Fine-scale surface normal estimation using a single NIR image”. In: *Proc. ECCV*. 2016.
- [180] Ye Yu and William A. P. Smith. “InverseRenderNet: Learning Single Image Inverse Rendering”. In: *Proc. CVPR*. 2019.
- [181] Lu Yuan, Jian Sun, Long Quan, and Heung-Yeung Shum. “Image deblurring with blurred/noisy image pairs”. In: *ACM Transactions on Graphics (TOG)* 26.3 (2007), p. 1.
- [182] Xiaoxing Zeng, Xiaojiang Peng, and Yu Qiao. “DF2Net: A Dense-Fine-Finer Network for Detailed 3D Face Reconstruction”. In: *Proc. ICCV*. 2019.



- [183] Jian Zhang and Bernard Ghanem. “ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing”. In: *Proc. CVPR*. 2018.
- [184] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. “Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising”. In: *IEEE Transactions on Image Processing* 26.7 (2017), pp. 3142–3155.
- [185] Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. “Learning deep CNN denoiser prior for image restoration”. In: *Proc. CVPR*. 2017.
- [186] Kai Zhang, Wangmeng Zuo, and Lei Zhang. “FFDNet: Toward a fast and flexible solution for CNN based image denoising”. In: *IEEE Transactions on Image Processing* (2018).
- [187] Lei Zhang, Xiaolin Wu, Antoni Buades, and Xin Li. “Color demosaicking by local directional interpolation and nonlocal adaptive thresholding”. In: *Journal of Electronic imaging* 20.2 (2011), p. 023016.
- [188] Xuaner Zhang, Jonathan T Barron, Yun-Ta Tsai, Rohit Pandey, Xiuming Zhang, Ren Ng, and David E Jacobs. “Portrait shadow manipulation”. In: *ACM Transactions on Graphics (TOG)* 39.4 (2020), pp. 78–1.
- [189] Yinda Zhang and Thomas Funkhouser. “Deep depth completion of a single rgb-d image”. In: *Proc. CVPR*. 2018.
- [190] Yulun Zhang, Kunpeng Li, Kai Li, Bineng Zhong, and Yun Fu. “Residual non-local attention networks for image restoration”. In: *arXiv preprint arXiv:1903.10082* (2019).
- [191] Ziyu Zhang, Alexander G Schwing, Sanja Fidler, and Raquel Urtasun. “Monocular object instance segmentation and depth ordering with cnns”. In: *Proc. ICCV*. 2015.
- [192] Hao Zhou, Sunil Hadap, Kalyan Sunkavalli, and David W. Jacobs. “Deep Single-Image Portrait Relighting”. In: *Proc. ICCV*. 2019.
- [193] Shangchen Zhou, Jiawei Zhang, Jinshan Pan, Haozhe Xie<sup>1</sup>, Wangmeng Zuo, and Jimmy Ren. “Spatio-Temporal Filter Adaptive Network for Video Deblurring”. In: *Proc. ICCV*. 2019.
- [194] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. “Unsupervised learning of depth and ego-motion from video”. In: *Proc. CVPR*. 2017.
- [195] Wei Zhuo, Mathieu Salzmann, Xuming He, and Miaomiao Liu. “Indoor scene structure analysis for single image depth estimation”. In: *Proc. CVPR*. 2015.
- [196] Magauiya Zhussip, Shakarim Soltanayev, and Se Young Chun. “Training deep learning based image denoisers from undersampled measurements without ground truth and without image prior”. In: *Proc. CVPR*. 2019.
- [197] Daniel Zoran, Phillip Isola, Dilip Krishnan, and William T Freeman. “Learning ordinal relationships for mid-level vision”. In: *Proc. CVPR*. 2015.
- [198] Daniel Zoran and Yair Weiss. “From learning models of natural image patches to whole image restoration”. In: *Proc. ICCV*. 2011.