NOISE MODELLING FOR SMARTPHONE CAMERAS

ABDELRAHMAN ABDELHAMED

A DISSERTATION SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN

ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

YORK UNIVERSITY

TORONTO, ONTARIO

November 2020

©ABDELRAHMAN ABDELHAMED, 2020

Abstract

In our everyday life we capture photographs. We create these images by measuring the amount of light, radiated from scenes in our physical world, on camera sensors embedded in our smartphones. Image noise is variation in the measurement of intensities or colours in digital images and it has the undesirable effect of obscuring information in images. Image noise is produced from two main sources: (1) the unavoidable, random nature of light and (2) the imaging sensor and associated circuitry. Unlike professional cameras, smartphone cameras have much smaller imaging sensors which makes them more susceptible to higher and more complex noise. To model, and ultimately remove, image noise, many mathematical models have been proposed. These models either represent synthetic noise or rely on assumptions that makes them unable to model real noise distributions observed from empirical data. One major reason for that is the lack of sufficient real noisy image datasets with ground truth images that can enable the study of real camera noise. The purpose of this dissertation is to provide a study on image noise modelling based on data-driven approaches specific to smartphone cameras. To this end, we first propose a systematic method for estimating ground truth noise-free images from noisy images captured by smartphone cameras. Using the proposed method, we collect a large-scale dataset, termed the Smartphone Image Denoising Dataset (SIDD), of high-quality images that can be used for noise modelling. Next, we utilize the SIDD dataset to devise a generative noise model, termed Noise Flow, that can be used to synthesise realistic noisy images to be utilized in many computer vision tasks. We also use our datatset to provide a benchmark for image denoising algorithms on real noisy images. As part of this benchmarking effort, we have developed an online image denoising challenge with the necessary software tools to facilitate the evaluation of image denoising methods applied to realistic noisy images. We believe the work in this dissertation helps to advance the state of the art in image noise modelling and image denoising.

To my parents and my wife

Acknowledgment

Praise be to Allah, the Most Gracious, the Most Merciful. Prayer and peace be upon the Prophet Muhammad. First of all, I would like to express my deepest gratitude and appreciation to my advisor *Prof. Michael S. Brown*, for his close guidance, great patience, endless support, and valuable advice throughout my whole study. Sincere thanks to my professors, Dr. Marcus Brubaker, Dr. Richard Wildes, Dr. Stephen (Steve) Lin, Dr. Scott MacKenzie, and my colleagues, Mahmoud Afifi, Abdullah Abuolaim, Dr. Hakki Karaimer, and Dr. Abhijith Punnappurath, and all my professors and colleagues at Lassonde School of Engineering, for their help and valuable discussions throughout this work. I would like to thank the whole Lassonde family for accepting me as a student and for providing me with resources to finish this work. Finally, I would like to express my heartful gratitude to my family, for their patience and support, words are useless in expressing my gratitude to my wife, my kids, and my parents, to whom I simply owe everything.

Abdelrahman
2020

Table of Contents

Abstract **ii**

Dedication **iii**

Acknowledgment **iv**

Table of Contents **v**

List of Tables **ix**

List of Figures **xi**

List of Acronyms **xvii**

Chapter 1: Introduction **1**

1.1 Contributions 5

1.2 Dissertation Outline 6

Chapter 2: Image Noise **8**

2.1 Photon Noise 8

2.2 Dark Current and Thermal Noise 9

2.3 Fixed Pattern Noise 10

2.4 Defective Photosites 11

2.5 Sensor Gain 11

2.6 Analog to Digital Conversion 12

2.7 Other Noise Types 14

Chapter 3: Image Noise Models **15**

3.1 Homoscedastic Models 16

3.1.1 Gaussian Assumption 16

3.1.2 Univariate Poisson Model 17

3.2 Heteroscedastic Models 17

3.2.1 Multivariate Poisson Model 17

3.2.2 Poisson-Gaussian Model 17

3.2.3 Heteroscedastic Gaussian Model 18

3.3 Other Models 19

Chapter 4: Image Denoising **23**

4.1 A Review of Image Denoising 23

4.2 Multi-Image Noise Estimation 35

4.3 Single-Image Noise Estimation 35

4.4 Denoising and Colour Spaces 38

4.5 Denoising and Synthetic Noise 39

4.6 Denoising Datasets and Benchmarks 40

Chapter 5: A High-Quality Denoising Dataset for Smartphone Cameras **43**

5.1 Motivation 43

5.2 Dataset 46

5.2.1 Image Capture Setup and Protocol 46

5.2.2 Noise Estimation 48

5.3 Ground Truth Estimation 50

5.3.1 Defective Pixel Correction 51

5.3.2 Intensity Alignment 52

5.3.3 Dense Local Spatial Alignment 53

5.3.4 Robust Mean Image Estimation 58

5.3.5 Quality of our ground truth versus the DND dataset 60

5.4 Benchmark 62

5.4.1 Setup 62

5.4.2 Results and Discussion 64

5.4.3 Application to CNN Training 64

5.5 Conclusion 67

Chapter 6: Noise Flow: Noise Modelling with Conditional Normalizing Flows 69

6.1 Motivation 70

6.2 Background and Related Work 71

6.2.1 Normalizing Flows 73

6.3 Noise Flow 76

6.3.1 Noise Modelling using Normalizing Flows 76

6.3.2 Noise Modelling using Conditional Normalizing Flows 77

6.4 Experiments 80

6.4.1 Experimental Setup 80

6.4.2 Results and Ablation Studies 81

6.5 Application to Real Image Denoising 85

6.6 Conclusion 86

Chapter 7: Benchmarking Denoising Algorithms on Real Noisy Images 93

7.1 Motivation 93

7.2 The Challenge 94

7.2.1 Dataset 94

7.2.2 Challenge Design and Tracks 95

7.3 Challenge Results 98

7.4 Conclusion 106

Chapter 8: Conclusion and Future Work 108

8.1 Conclusion 108

8.2 Future Research Directions 110

8.2.1 Realistic Noise Models in Perceptual Colour Spaces 110

8.2.2 Towards Accurate Noise Estimation 111

8.2.3 Towards Better Image Denoising 112

Bibliography **113**

Appendices **131**

Appendix A: List of Publications 131

List of Tables

3.1 A brief summary of the types of noise that are *explicitly* modelled be each of the discussed noise models. 22

5.1 Details about the smartphone cameras we used in capturing our dataset and the number of scene instances and images captured by each camera. 47

5.2 Denoising performance in terms of PSNR (dB), SSIM, and denoising time in seconds per 1-Mpixel image (1024×1024 pixels) for the benchmarked methods averaged over 40 images. 62

5.3 Details on the two subsets of RAW image patches we used for training the DnCNN denoising model. σ_{min}, σ_{max}, and σ_{μ} indicate the minimum, maximum, and mean noise levels in the subsets. 65

5.4 Mean noise estimates (β_1, β_2, and σ) of the denoised testing image patches using the four DnCNN models trained on subsets A and B. Training on our ground truth with real noise mostly yields higher quality images. . . 65

6.1 Best achieved testing NLL and marginal D_{KL} for Noise Flow compared to the Gaussian and Camera NLF baselines. Relative improvements of Noise Flow on other baselines, in terms of *likelihood*, are in parentheses. 82

6.2 Best achieved testing NLL and marginal D_{KL} for different layer architectures. 84

6.3 DnCNN denoiser [125] trained on synthetic noise generated with Noise Flow (DnCNN-NF) achieves higher PSNR and SSIM values compared to training on synthetic noise, from a Gaussian model or camera NLFs, and real noise. 87

7.1 Results and rankings of methods submitted to the 2019 raw-RGB denoising track. 103

7.2 Results and rankings of methods submitted to the 2019 sRGB denoising track. 104

7.3 Results and rankings of methods submitted to the 2020 rawRGB denoising track. 105

7.4 Results and rankings of methods submitted to the 2020 sRGB denoising track. 106

List of Figures

1.1 A typical relationship between the three research areas: image noise modelling, estimation, and reduction (i.e., denoising). This dissertation is focused on the highlighted areas: estimating ground truth images and modelling image noise. 4

2.1 A simplified model of an imaging pipeline showing (bottom row) imaging processes and (top row) the associated noise types. Model adapted from [53, 60, 64, 82]. 9

2.2 A simple illustration of the signal-dependency of photon noise. The higher the signal (i.e., intensity), the higher the standard deviation of noise (σ). . 9

2.3 A simple illustration of dark current noise and thermal noise. With increasing integration time from two to four minutes, the thermal effect on the dark current noise can be observed. 10

2.4 A simple illustration of defective pixels. On the left, a noisy image. On the right, an average of 150 observations of the image on the left. The averaging process reveals the locations of some hot and stuck pixels. . . 12

2.5 A simple illustration of sensor gain and ISO level showing an image captured under different ISO levels. It is observable that the higher the ISO, the higher the noise. 13

2.6 (Left) An ideal analog-to-digital converter's (ADC) transfer function. (Right) An actual ADC's transfer function. Figure reproduced from [72]. 14

3.1 (Top) In-camera imaging pipeline. (Bottom) Changes in noise distribution through the stages of the camera imaging pipeline. Figure reproduced from [94]. 20

4.1 Illustration of matching blocks from noisy images corrupted by white Gaussian noise with standard deviation 15 and zero-mean. Each image shows a reference block marked with "R" and a few of the blocks matched to it. Figure reproduced from [32]. 32

4.2 Pixels $q1$ and $q2$ have a large weight because their neighbourhood windows are similar to that of pixel p. On the other side, the weight of pixel $q3$ would be much smaller because the similarity between windows is lower. 33

4.3 The distribution of the highly cited 980 papers with the phrase "image denoising" in the title, over the years 1995 – 2017. The number of citations of the papers from each are also shown. Data was collected from Google Scholar. 34

4.4 An example image illustrating the smooth image model. (Left) An input image. (Right) The image partitioned into piece-wise smooth regions using a simple K-Means clustering method for grouping pixels into regions, as described in [130]. 36

4.5 (Left) A synthesised noisy image and (right) its corresponding NLF (noise standard deviation as a function of image intensities). The red, green, and blue curves are estimated using the algorithm from [82], whereas the gray curves are the true values for the synthetically generated noise. 37

4.6 (Left) A noisy sRGB image. (Right) The selected low-rank image patches for estimating the signal-dependent noise parameters using the method from [85]. Figure reproduced from [85]. 38

5.1 An example of a scene imaged with an *LG G4* smartphone camera: (a) a high-ISO noisy image; (b) the same scene captured with low ISO – this type of image is often used as ground truth for (a); (c) ground truth estimated by [99]; (d) our ground truth. 45

5.2 Twenty example scene instances from our dataset. The labels indicate, in the following order: the scene instance number, the scene number, the camera ID, ISO level, exposure time (seconds), illumination temperature (K), and lighting condition (low-light or normal-light). The camera IDs are indicated in Table 5.1. 48

5.3 Examples of noisy images from our dataset captured under different lighting conditions and camera settings. Below each scene, zoomed-in regions from both the noisy image and our estimated ground truth (more details in Section 5.3) are provided. 49

5.4 A block diagram illustrating the main steps in our procedure for ground truth image estimation. 51

5.5 An example of a mean image (c) computed over a sequence of low-light images where defective pixels are present, and our corresponding ground truth (d) where defective pixels were corrected. One of the images from the sequence is shown in (a) and zoomed-in in (b). 52

5.6 (a) A part of a static planar chart with fiducials imaged on a vibration-free optical table. The quiver plots of the observed and measured pixel drift between the first and last (500^{th}) image in a sequence of 500 images are shown for (b) iPhone 7 and (c) Google Pixel. (d) The effect of replacing our local alignment technique by a global 2D translation to align a sequence of images after synthesizing the local pixel drift from (b). 53

5.7 Local pixel shifts throughout a sequence of 500 images as indicated from a planar chart of fiducials. We plot the pixel shifts along the x-axis and y-axis at the five locations marked by green squares. We show results for Google Pixel, Motorola Nexus 6, and Apple iPhone 7. 56
5.8 The first column shows the local pixel shift between the 1st and the 500th frames of an image sequence of the fiducials chart shown in Figure 5.7. The second and third columns show the interpolation of local pixel shifts using 2D thin-plate splines. We show results for Google Pixel, Motorola Nexus 6, and Apple iPhone 7. 57
5.9 Comparison between methods used for estimating the mean image (a) over a range of number of images and (b) over a range of the first parameter of signal-dependent noise (β_1). The adopted method, WLS fitting of the CDF with censoring, yields the lowest MSE. 60
5.10 (a) Comparison between noise levels in our ground truth images versus the ground truth estimated by [99] for five scenes. Our ground truth has lower noise levels in four out of five images. (b) Comparison of noise levels in our dataset versus DND dataset. 61
5.11 Testing results of DnCNN [125] using two types of ground truth (post-processed low-ISO images and our ground truth images) and two types of noise (synthetic and real) on two randomly selected subsets of our dataset (shown in Table 5.3). Training using our ground truth images on real noise yields the highest PSNRs. 67
5.12 The effect of using various synthetic noise levels (σ) to train DnCNN [125] on two randomly selected subsets of patches from our dataset, as shown in Table 5.3. The mean noise levels σ_μ on subsets A and B are 2.62 and 9.73, respectively. 68

6.1 Synthetic noisy images generated by (a) a Gaussian model, (b) a heteroscedastic signal-dependent model represented by camera noise level functions (NLF), and (c) our Noise Flow model. Synthetic noise generated from Noise Flow is consistently the most similar to the real noise in (d), qualitatively and quantitatively. (e) Reference clean image. 71

6.2 The architecture of our Noise Flow model. The affine coupling and 1×1 convolutional layers are ported from [75]. The signal-dependent and gain layers are newly proposed. The Raw-to-sRGB pipeline is ported from [3]. 76

6.3 (a) NLL per dimension on the training and testing sets of Noise Flow compared to (1) the Gaussian model and (2) the signal-dependent model as represented by the camera-estimated NLFs. (b) Marginal KL divergence (D_{KL}) between the generated and the real noise samples. 82

6.4 Generated noise samples from (c) Noise Flow are much closer, in terms of marginal KL divergence, to (d) the real samples; compared to (a) Gaussian and (b) camera NLF models. (e) Clean image. Corresponding ISO levels and lighting conditions are on the left. 88

6.5 (a) Signal-dependent noise parameters β_1 and β_2 are consistent with the signal-dependent noise model where β_1 is dominant and β_2 is much smaller. (b) The gain parameters, in log scale, are consistent with the corresponding ISO levels shown in the legend. 89

6.6 (a) Learned camera-specific weights for the shared gain layer indicates differences in the gain of different cameras. These gains are correlated with the cameras' different values for NLF parameter β_1 shown in (b). The iPhone and G4 cameras have smaller ranges of ISO values and hence their correlation with the gains is not clear. 89

6.7 Sample denoising results from DnCNN trained on three different noise synthesis methods: (b) Gaussian; (c) camera NLF; and (d) Noise Flow. (e) DnCNN trained on real noise. (a) Real noisy image. (f) Ground truth. . . 90

6.8 Results of training the DnCNN [125] image denoiser with synthetic noise generated by: the Gaussian model; the camera NLFs; and our Noise Flow model. 91

6.9 Testing PSNR results corresponding to the three model from Figure 6.8. The DnCNN model trained on noise generated with Noise Flow yields the best PSNRs. 91

6.10 Sample denoising results from the DnCNN denoiser trained on three different noise synthesis methods:(b) Gaussian; (c) camera NLF; and (d) Noise Flow. (a) Real noisy image. (e) Ground truth. 92

7.1 A block diagram of the simulated camera rendering pipeline provided with the NTIRE challenge. 98

7.2 Combined PSNR and SSIM values of method from the 2019 raw-RGB track. 99

7.3 Combined PSNR and SSIM values of method from the 2019 sRGB track. . 100

7.4 Combined PSNR and SSIM values of method from the 2020 rawRGB track. 100

7.5 Combined PSNR and SSIM values of method from the 2020 sRGB track. . 101

7.6 Combined PSNR and SSIM values of all methods from both raw-RGB (in blue) and sRGB (in red) tracks from the 2019 challenge. Note the different axes and scales for each track. 101

7.7 Combined PSNR and SSIM values of all methods from both rawRGB (in blue) and sRGB (in red) tracks from the 2020 challenge. Note the different axes and scales for each track. 102

List of Acronyms

AWGN	Additive White Gaussian Noise
BM3D	Block Matching and 3D Transform
BSD	Berkeley Segmentation Dataset
CDF	Cumulative Distribution Function
CNN	Convolutional Neural Network
DCT	Discrete Cosine Transform
DnCNN	Denoising Convolutional Neural Network
DND	Darmstadt Noise Dataset
DNG	Digital Negative
DNN	Deep Neural Network
DSIMM	Structural Dissimilarity
DSLR	Digital Single-Lens Reflex
DSNU	Dark Signal Non-Uniformity
EPLL	Expected Patch Log Likelihood
FoE	Fields of Experts
FPN	Fixed Pattern Noise
GAN	Generative Adversarial Network
GLIDE	Global Image Denoising
ISP	Image Signal Processor
KL	Kullback-Leibler

KSVD	K-means Singular Value Decomposition
LPG	Local Pixel Grouping
MLE	Maximum Likelihood Estimation
MLP	Multi-Layer Perceptron
MS-SSIM	Multi-Scale Structural Similarity
NAS	Neural Architectural Search
NF	Noise Flow
NLF	Noise Level Function
NLM	Non-Local Means
NTIRE	New Trends in Image Restoration and Enhancement
OIS	Optical Image Stabilization
PCA	Principal Component Analysis
PRNU	Photo Response Non-Uniformity
PSNR	Peak Signal-to-Noise Ratio
RMSE	Root Mean Squared Error
SIDD	Smartphone Image Denoising Dataset
SSIM	Structrual Similarity
SVD	Singular Value Decomposition
TNRD	Trainable Nonlinear Reaction Diffusion
WLS	Weighted Least Squares
WNNM	Weighted Nuclear Norm Minimization

Chapter 1

Introduction

We now have the ability to take photographs at almost anytime and anyplace using our smartphones. These images are created by recording the light measurements, radiated from scenes in our physical world, using cameras or other imaging devices. The light measurement devices can be chemical, such as the classic photographic films, or electronic, such as digital image sensors.

Digital image sensors have become the dominant devices used in photography, hence, the term digital photography. Digital image sensors typically consist of arrays of electronic photosites (also known as photodetectors or photosensors). When a series of photons is emitted or reflected from an object and hits a photosite, a proportional series of electrons is produced resulting in an electric charge. This electric charge is then measured and converted to a digital value. These digital values are referred to as intensities or brightness values. When an image is captured, intensities of the arrays of photosites are measured, digitised, and stored as pixels (i.e., picture elements) in digital files ready for further digital processing, viewing, publishing, or printing.

From the instant when we press a button to capture an image, until the image is stored as a digital file, many processes are taking place — some processes have desirable effect, others do not. Some of the mostly undesired effects are collectively referred to as *image* *noise*. Image noise usually results in obscuring information in images, such as structures, textures, and intensity or colour information. Image noise, in its most common form, is the result of a random process, or a combination of random processes, introducing variation in the measurement of intensities or colour information in digital images; however, image noise could also arise from other sources that are not inherently random, such as the defective photosites found in imaging sensors due to some manufacturing imperfections. Image noise is typically produced from two main sources: (1) the unavoidable particle nature of light and (2) the imaging sensor and the associated circuitry of a digital camera. The first source of image noise, the particle nature of light, produces what is referred to as photon noise (an instance of shot noise). Photon noise exists because light emitted or reflected from an object consists of discrete (i.e., quantized) packets of energy (i.e., photons). Physically, the number of photons emitted from a scene over a unit time is random. This randomness results in fluctuations in the number of photons hitting a photosite over a unit of time. Such fluctuations are termed *photon noise*. The second source of image noise, the sensor and circuitry, produces a combination of noise types, such as dark current noise, sensor read-out noise, quantization noise, cross-talk noise, fixed pattern noise, defective pixels noise, etc. Typically, image noise is measured in terms of a statistical variance of some underlying distribution. Some of the aforementioned noise types will be discussed later in Chapter 2.

The fact that our photographs are corrupted by image noise has led to the development of a prolific number of image denoising methods over the past few decades, growing and expanding the field of image denoising (also known as image noise reduction). However, image denoising is tightly coupled with two other processes: noise modelling and noise estimation. A denoising method normally relies on some theoretical model of the noise that needs to be reduced. Also, image denoising typically involves, either explicitly or implicitly, the estimation of noise level in the image to be denoised.

To characterise the various noise types occurring in the imaging pipeline, many mathematical models have been proposed in the literature. The simplest and most common model is the Gaussian noise model [15, p. 149], or more precisely the homoscedastic Gaussian noise model. Under this model, the noise values are Gaussian-distributed, and all image pixels share the same noise variance. Due to its simplicity, the Gaussian noise model has been widely used in the research literature. However, it has been shown that many noise types cannot be precisely modelled as Gaussian, such as the fixed-pattern noise and the photon noise in low-light environments [59].

Beyond the homoscedastic Gaussian assumption, more complex image noise models have been proposed to model different combinations of noise types. Such models include the noise level function (NLF) [81, 82] and the signal-dependent noise model (also known as the Poisson-Gaussian model [47]). Such models avoid the homoscedasticity assumption, instead they rely on the heteroscedasticity of the noise distribution over image pixels, in other words, different image pixels can have different noise variances. Image noise models will be reviewed in more detail in Chapter 3.

Once a noise model is decided on, a noise estimation process has to be applied to determine the parameter(s) of the noise model (e.g., the variance of a Gaussian distribution of the noise) either for a single image or a set of images. This led to the research focus on proposing methods for image noise estimation based on the various noise models. However, noise estimation and image denoising can be thought of as chicken-and-egg problems. To estimate the noise in an image, a model of the underlying clean image needs to be assumed, which in turn could be obtained through applying a denoising method on the image; conversely, most denoising methods require some estimate of the noise as an input. Therefore, noise estimation methods are typically coupled with noise modelling or image denoising methods (e.g., [47, 81, 82]). It is worth noting that better noise estimation may lead to better denoising performance; however, both noise estimation and denoising are restricted by the accuracy of the underlying noise model. More details about noise estimation methods will be provided in Chapter 4.

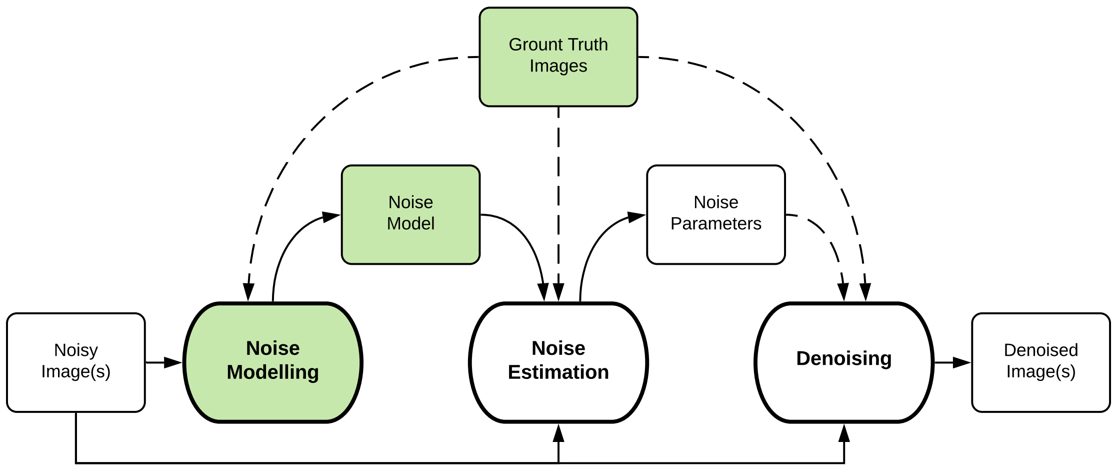


Figure 1.1: A typical relationship between the three research areas: image noise modelling, estimation, and reduction (i.e., denoising). This dissertation is focused on the highlighted areas: estimating ground truth images and modelling image noise.

After establishing the noise model and having an estimation of the noise present in an image, what follows is the image denoising process. Image denoising is a fundamental step in any imaging system for improving image quality. It refers to the process of minimising, and ideally removing, noise from images. Typically, image denoising is based on a specific noise model and involves the estimation of that model's parameters. Since there is a massive number of denoising methods available, the focus of this review will be on the common strategies followed in developing image denoising methods, with some focus on some of the most prominent and state-of-the-art denoising methods. A review on image denoising strategies will be provided in Chapter 4.

To this introductory end, the three research areas mentioned above (i.e., image noise modelling, estimation, and reduction) can be tied together as shown in Figure 1.1. Given observations of noisy images, noise modelling aims at figuring out the mathematical model that best fits the noise distribution found in the images. Then, noise estimation aims at determining the parameters of a given noise model from one or more noisy images. Finally, image denoising aims at reducing or removing noise from such images based on a given noise model and, optionally, the model parameters. For all three processes, performance evaluation typically requires having the ground truth image or some proxy to noise-free images.

1.1 Contributions

This dissertation provides three research contributions summarized as follows. First, we establish a much-needed image dataset for smartphone image denoising research. We propose a systematic procedure for estimating ground truth for real noisy images that can be used to benchmark denoising performance on smartphone imagery. Using this procedure, we captured a the Smartphone Image Denoising Dataset (SIDD) of $\sim 30,000$ real noisy images using five representative smartphone cameras and generated their ground truth images. Using our dataset, we benchmark a number of denoising methods to gauge the relative performance of various approaches, including patch-based methods and more recent CNN-based techniques. From the provided analysis, we show that for CNN-based methods, notable gains can be made when using our ground truth data versus conventional alternatives such as low-ISO images.

Second, we develop a novel noise model (Noise Flow) that combines the insights of parametric noise models and the expressiveness of powerful generative models. Specifically, we leverage recent normalizing flow architectures [75] to accurately model noise distributions observed from large datasets of real noisy images. In particular, based on the recent Glow architecture [75], we construct a normalizing flow model which is conditioned on critical variables, such as intensity, camera type, and gain settings. The model can be shown to be a strict generalization of the camera-specific noise models but with the ability to capture significantly more complex behaviour. The result is a single model that is compact (fewer then 2500 parameters) and considerably more accurate than existing models. To demonstrate the effectiveness of Noise Flow, we consider the application of denoising and use Noise Flow to synthesize training data for image denoising resulting in significant improvements in image quality.

Lastly, using our SIDD dataset, we provide the research community with a comprehensive framework for benchmarking image denoising algorithms. As part of this framework, we developed an image denoising challenge with the goal of gauging and advancing the state of the art in image denoising with more focus on *real*, rather than synthetic, noisy images. In this challenge, we designed two tracks for benchmarking image denoisers on both raw sensor data (raw-RGB) and standard RGB (sRGB) colour spaces. We also developed essential tools to facilitate the development of image denoisers in the raw-RGB color space. Such tools included a simulated camera rendering pipeline and camera metadata extraction tools. This challenge has been hosted twice in the workshop on New Trends in Image Restoration and Enhancement (NTIRE), in 2019 and 2020.

We believe our study including the developed methods, datasets, and noise models will be useful in advancing image denoising and other computer vision tasks for images captured with smartphones. All developed methods and source codes have been made publicly available to the research community. (Links are provided in Appendix 8.2.3.)

1.2 Dissertation Outline

In this dissertation, Chapters 2 and 3 present a review of the noise sources and models used for digital imaging sensors. Chapter 4 presents a review on image denoising and noise estimation methods with some focus on the common research practices in the field. Throughout these chapters, discussions are provided, and research gaps are identified. Then, in Chapter 5, we present our approach to generating high-quality noise datasets with ground truth data and how we used this approach to generate the SIDD. In Chapter 6, we present our approach to data-driven noise modelling via deep generative models and how we developed the Noise Flow model. In Chapter 7, we discuss the development of our SIDD benchmark for real image denoising and how it is used to gauge the state of the art in image denoising. Finally, in Chapter 8, we summarise our study and contributions, then open a discussion for further research direction for advancing research in the areas of image noise modelling and denoising.

Chapter 2

Image Noise

Image noise emerges from many sources. Such sources can be divided into two main categories based on the nature and timing of the process generating the noise: (1) physical sources associated with the quantum nature of light and unrelated to the imaging sensor; and (2) electronic sources associated with the imaging sensor and circuitry. The noise associated to the imaging sensor arise from common processes happening on most sensors [49], such as detection and conversion of photons to electrons, readout of the electric charges, signal amplification or gain, analog-to-digital conversion, colour processing, and interface electronics. Some of these noise types are shown in Figure 2.1 along with their associated imaging processes. In the following, some of the common noise sources and types are discussed.

2.1 Photon Noise

Photon noise (an instance of shot noise) is the fluctuations in the measured amount of light (number of photons) hitting a photosensor. The source of such fluctuations is the discrete nature of light, where the number of photons measured in a unit time follows a random distribution. The Poisson distribution is commonly used to describe photon noise [59]. The probability of counting a number of photons k over an integration time

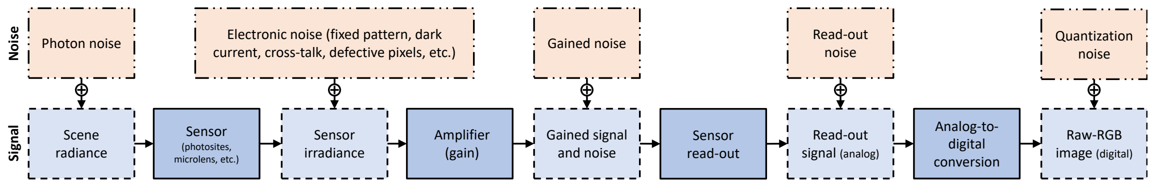


Figure 2.1: A simplified model of an imaging pipeline showing (bottom row) imaging processes and (top row) the associated noise types. Model adapted from [53, 60, 64, 82].

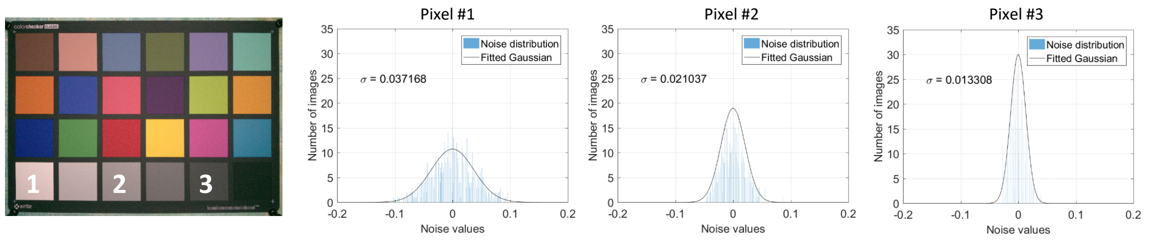


Figure 2.2: A simple illustration of the signal-dependency of photon noise. On the left, an image captured 500 times under fixed conditions. Three pixel locations having different intensities are indicated by numbers. On the right, the noise distribution at the specified three pixels over 500 observations of the image on the left. The higher the signal (i.e., intensity), the higher the standard deviation of noise (σ).

interval t is represented by a random variable N with probability function

$$p(N = k) = \frac{e^{(-\lambda t)}(\lambda t)^k}{k!}, \tag{2.1}$$

where λ is the expected number of photons per unit time interval.

From Equation 2.1, it appears that photon noise is signal-dependent, i.e., the variance of the signal is equal to its expectation $\mathrm{Var}[N] = \mathrm{E}[N] = \lambda t$, and hence, the standard deviation of the photon noise is proportional to the square root of the signal.

2.2 Dark Current and Thermal Noise

Dark current noise is caused by the randomly generated electrons that are independent of the light signal. Such electrons are thermally generated even in the absence of a light signal. Similar to photon noise, dark current noise is also related to the integration time

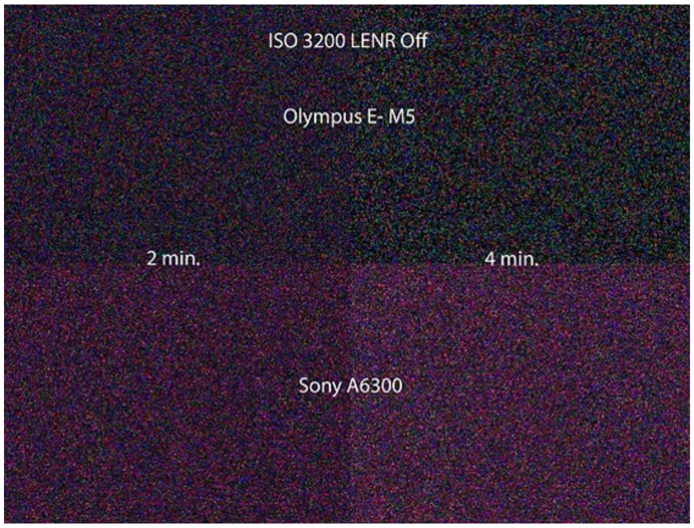


Figure 2.3: A simple illustration of dark current noise and thermal noise. The images shown are dark frames, i.e., captured in a no-light environment, showing the dark current noise. With increasing integration time from two to four minutes, the thermal effect on the dark current noise can be observed.

interval and also follows a Poisson probability distribution. Another factor that contributes to the dark current noise is heat, i.e., the temperature of the sensor, where the hotter the sensor element gets, the higher the dark current noise. This is illustrated in Figure 2.3. Dark current noise is a major contributor to the fixed-pattern noise (FPN) discussed next.

2.3 Fixed Pattern Noise

Fixed pattern noise (FPN) is a general term that represents non-uniformity in the behaviour of different photosites in an imaging sensor. Two main sources give rise to fixed pattern noise, commonly known as: dark signal non-uniformity (DSNU) and photo response non-uniformity (PRNU) [30, 40]. On one hand, DSNU refers to the non-uniformity in the dark current noise of individual photosites, i.e., different photosite may generate different levels of dark current noise under the same conditions (e.g., heat, integration time, etc.). On the other hand, PRNU refers to the non-uniformity in the photon noise generated by individual photosites, i.e., different photosites may generate different photon noise levels under the same conditions, including the amount on incident light.

2.4 Defective Photosites

Defective photosites (commonly referred to as defective pixels) are sensor elements whose behaviour deviates from the one of an average sensor element, i.e., a normal photosite [66, 67]. Such deviation is usually due to manufacturing imperfections. Defective photosites may be categorised into three types, commonly referred to as: dead, stuck, and hot photosites.

Dead photosites typically do not produce any signal, or an extremely weak signal compared to normal photosites, subsequently, they result in black pixels in the sensor's raw output or pixels of standout colours in the processed images. Dead photosites can be easily masked during the manufacturing process using hardware or software components.

Stuck photosites produce too large signals such that they can be easily saturated, resulting in too-bright or over-saturated pixels in the output images. On the other hand, hot photosites produce signals that are not saturated, but noticeably higher than signals produced by normal photosites. This happens due to their too-high sensitivity to light and/or heat, resulting in pixels that are noticeably brighter than expected. A simple illustration of defective pixels is shown in Figure 2.4.

2.5 Sensor Gain

To amplify weak light signals, sensor gain is applied. Sensor gain could be analog or digital, according to whether it is applied before or after the analog-to-digital conversion (discussed later). Sensor gain is a factor g that represents the ratio of the actual measured signal and the sensor's readout signal. For example, $g = 0.5$ means that the sensor's

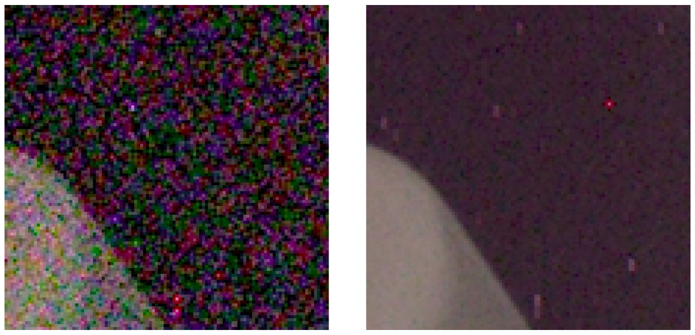


Figure 2.4: A simple illustration of defective pixels. On the left, a noisy image. On the right, an average of 150 observations of the image on the left. The averaging process reveals the locations of some hot and stuck pixels.

readout signal is double the actual measured signal. This is inversely proportional to the commonly-known ISO level G [60] where the relationship is

$$G = \frac{U}{g}, \tag{2.2}$$

where U is a sensor-dependent constant.

Amplifying the image signal means also amplifying the associated noise occurring before the gain application, as shown in Figure 2.5. Despite that applying sensor gain, at least in theory, does not change the signal-to-noise ratio of the image, it can have larger effect on the later stages on-board the sensor, such as analog-to-digital conversion and sensor readout.

2.6 Analog to Digital Conversion

Another major process happening on-board digital imaging sensors is the analog-to-digital conversion, or digitisation, of the signal. This process introduces two main types of noise into the image signal: quantization noise and input-referred noise [72, 128, 54].

An analog-to-digital converter (ADC) converts an analog (i.e., continuous) signal into a digital (i.e., discrete) representation, in what is referred to as a quantization process [55].

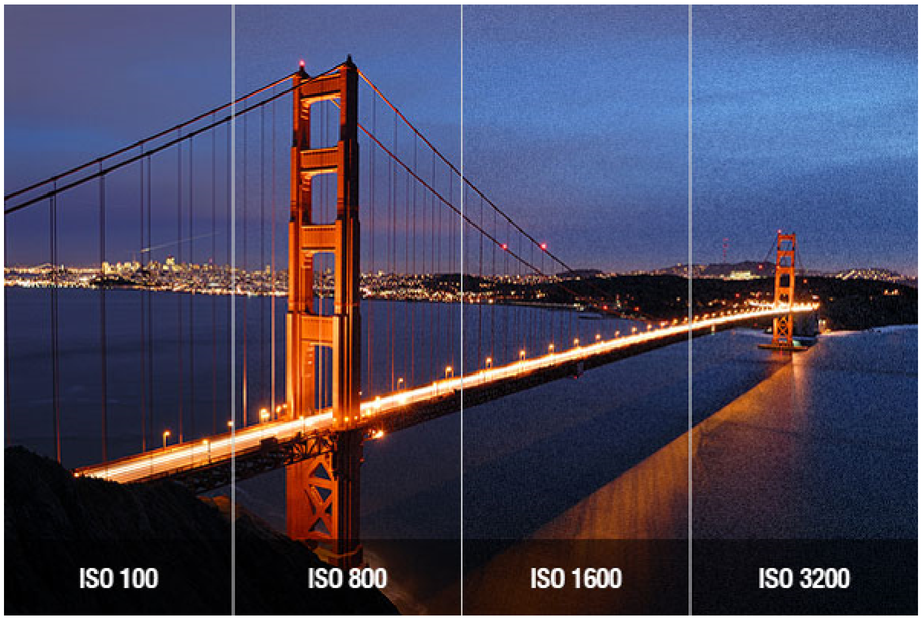


Figure 2.5: A simple illustration of sensor gain and ISO level showing an image captured under different ISO levels. It is observable that the higher the ISO, the higher the noise.

Quantization means that a range of input values are collapsed into the same output value. That range is called quantum (q) and is equivalent to the least significant bit (LSB) of the output digital representation. The difference between input and output is called the quantization error. Therefore, the quantization error can be between $\pm q$.

Input-referred noise, also known as code-transition noise, is basically an internal resistor noise that leads to deviations from the ideal digital code expected as an output from ADCs. As the analog input voltage is increased, an *ideal* ADC maintains a constant output code until a transition region is reached, at which point it instantly jumps to the next value, remaining there until the next transition region is reached. A theoretically perfect ADC has zero code-transition noise, and a transition region width equal to zero. A practical ADC has a certain amount of code-transition noise, and therefore a finite transition region width, as shown in Figure 2.6.

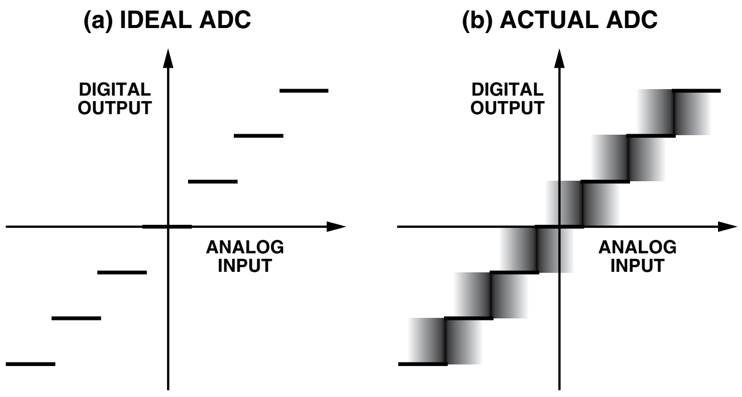


Figure 2.6: (Left) An ideal analog-to-digital converter's (ADC) transfer function. (Right) An actual ADC's transfer function. An ideal ADC maintains a constant output code until a transition region is reached. A practical ADC has a certain amount of code-transition noise (i.e., input-referred noise). Figure reproduced from [72].

2.7 Other Noise Types

The aforementioned sources and types of noise are the most common in the research literature. However, there are other noise sources and types on-board an imaging sensor. Examples include cross-talk and clipping noise. Cross-talk [49, 65] is a term referring to the noise resulting when photons falling on one photosite are falsely sensed by other neighbour photosites. Clipping noise [46, 97] is caused by the limits of the intensity dynamic range of the imaging sensor. When the intensity dynamic range of the imaged scene is larger than the sensor's range, some scene intensities cannot be measured by the sensor. This happens in two possible ways: under-exposure and over-exposure. Under-exposure happens when a scene pixel's intensity is lower than the black level of the sensor. Conversely, over-exposure happens when a scene pixel's intensity is higher than the saturation level of the sensor.

With the above discussion of the various sources and types of image noise, it becomes clear that the better we understand these various noise types, the better we can model, estimate, and reduce them. In Chapter 3, existing image noise models commonly used in the literature are discussed in detail.

Chapter 3

Image Noise Models

Image noise can be described as deviations of the measurements from the actual signal and results from a number of causes, including physical phenomena, such as photon noise, or by the electronic characteristics of the imaging sensors, such as fixed pattern noise. Given an observed image $\tilde{\mathbf{I}}$ and its underlying noise-free image $\mathbf{I}$, their relationship can be written as

$$\tilde{\mathbf{I}} = \mathbf{I} + \mathbf{n}, \tag{3.1}$$

where $\mathbf{n}$ is the noise corrupting $\mathbf{I}$. The focus of noise modelling research is to model the noise $\mathbf{n}$. Equation 3.1 represents a commonly used form of noise, additive noise. Other forms of noise can be used as well, such as multiplicative noise which can be represented as

$$\tilde{\mathbf{I}} = \mathbf{I}\,\mathbf{n}_{\mathrm{m}}, \tag{3.2}$$

where $\mathbf{n}_{\mathrm{m}}$ is a multiplicative noise factor. Multiplicative noise can still be represented as additive noise with the noise component represented as a multiplication function of the noise-free image as $\tilde{\mathbf{I}} = \mathbf{I} + (\mathbf{n}_{\mathrm{m}} - 1)\,\mathbf{I}$.

Image noise models can be roughly divided into two main categories: homoscedastic and heteroscedastic. Homoscedastic noise models are based on the assumption that noise in an image follows a univariate distribution and the noise values are independent and identically distributed. Conversely, heteroscedastic noise models are based on the assumption that noise variance is actually signal-dependent and is related to the image intensity. This section provides a detailed discussion of noise models from both categories.

3.1 Homoscedastic Models

The simplest and most practical noise models are homoscedastic. This is based on the assumption that the noise variance is the same at all image pixels. Such noise models are usually represented by univariate distributions.

3.1.1 Gaussian Assumption

The most common noise model is the Gaussian model. Such a model is still used in many recent works in image processing and computer vision (e.g., [25, 113, 125, 79]). Under the homoscedastic Gaussian assumption, also known as the additive white Gaussian noise (AWGN), the distribution of noise in an image is a Gaussian distribution with independent and identically distributed values:

$$n_i \sim \mathcal{N}(0, \sigma^2), \tag{3.3}$$

where n_i is the noise value at pixel i and follows a normal distribution with zero-mean and σ^2 variance. Despite its prevalence, the Gaussian model does not represent the fact that photon noise is signal-dependent, as discussed in Section 2.1.

3.1.2 Univariate Poisson Model

As an approximation of the signal dependency of noise, a univariate Poisson distribution can be used instead of a Gaussian distribution:

$$n_i \sim \mathcal{P}(\mathbf{I}_\lambda), \qquad \mathbf{I}_\lambda = \frac{1}{N}\sum_{i=1}^{N} \mathbf{I}_i \tag{3.4}$$

where $\mathbf{I}_\lambda$ is the mean image intensity and serves as the variance of the noise as well. Despite that the univariate noise models are simple and practical in many situations, they are not sufficient for accurate modelling of complex combinations of the various noise types discussed in Chapter 2.

3.2 Heteroscedastic Models

3.2.1 Multivariate Poisson Model

To account for the per-pixel signal dependency of photon noise (Section 2.1), a Poisson distribution $\mathcal{P}$ is used instead:

$$n_i \sim \alpha\mathcal{P}(\mathbf{I}_i) - \mathbf{I}_i, \tag{3.5}$$

where $\mathbf{I}_i$, the underlying noise-free intensity at pixel i, is both the mean and variance of the noise at that pixel, and α is a sensor-specific scaling factor of the signal. Despite that the multivariate Poisson model can sufficiently account for the dependency of noise on the individual pixel intensities in an image, it still cannot represent other sensor noise types, such as dark current noise and amplification noise, discussed in Chapter 2.

3.2.2 Poisson-Gaussian Model

Neither the Gaussian nor the Poisson models alone can accurately describe image noise. That is because image noise consists of both signal-dependent components and signal-independent components. To address such limitation, a Poisson-Gaussian model has been widely adopted [46, 47, 90, 88], where the image noise is a combination of a signal-dependent Poisson distribution and a signal-independent Gaussian distribution:

$$n_i \sim \alpha\, \mathcal{P}(\mathbf{I}_i) - \mathbf{I}_i + \mathcal{N}(0, \delta^2), \tag{3.6}$$

where δ^2 is a scalar that collectively represent the variance of all signal-independent noise components.

3.2.3 Heteroscedastic Gaussian Model

A more widely accepted alternative to the Poisson-Gaussian model is to replace the Poisson component by a Gaussian distribution whose variance is signal-dependent [85, 93], which is referred to as the heteroscedastic Gaussian model:

$$n_i \sim \mathcal{N}(0, \alpha^2\, \mathbf{I}_i + \delta^2). \tag{3.7}$$

The heteroscedastic Gaussian model is more commonly referred to as the **noise level function** (NLF) [82] and describes the relationship between image intensity and noise variance:

$$\operatorname{var}(n_i) = \beta_1\, \mathbf{I}_i + \beta_2, \qquad \beta_1 = \alpha^2, \beta_2 = \delta^2, \tag{3.8}$$

where β_1 and β_2 are commonly referred to as the NLF parameters and some cameras report them within the captured image files [3, 99].

Heteroscedastic signal-dependent models may accurately describe noise components such as photon noise and dark current noise. However, in real images there are still other noise sources that may not be accurately represented by such models [3, 46, 99]. Examples of such sources include fixed-pattern noise, defective pixels, clipped intensities, spatially correlated noise (i.e., cross-talk), amplification and quantization noise.

3.3 Other Models

Some attempts have been made to close the gap between the prior models and the realistic cases of image noise.

Clipped heteroscedastic model Image intensities may be clipped due to the limits of the sensor's dynamic range of intensities. A sensor cannot measure an intensity lower than its black level or higher than its saturation level. Such sensor's dynamic range is usually normalised, i.e., mapped to the range $[0, 1]$, for mathematical convenience. A clipped intensity is measured by the sensor as either zero or 1, based on whether it was originally lower than the sensor's dark level or higher than its saturation level, respectively. A clipped heteroscedastic model is needed [46] to account for the clipping effects on noise variance of the clipped pixels.

Assuming an observed clipped noisy pixel is

$$\mathbf{I}^{\mathrm{c}}{}_i = \min\{\max\{\tilde{\mathbf{I}}_i, 0\}, 1\}, \tag{3.9}$$

the corresponding clipped heteroscedastic noise would follow a doubly censored normal distribution [29]

$$n_i^{\mathrm{c}} \sim \mathcal{N}(0, \alpha^2\, \mathbf{I}_i + \delta^2) \quad \text{supported on} \quad [-\mathbf{I}_i, 1 - \mathbf{I}_i]\,. \tag{3.10}$$

Cross-channel model In most noise models, colour channels (i.e., red, green, and blue) are considered as mutually independent, especially in camera raw-RGB images, which are considered the immediate sensor output. However, signals from different colour channels get mixed during the imaging process inside the camera due to colour processes applied on them, such as colour gamut mapping, tone-mapping, and compression. To account for these cross-channel mix-ups, a noise model needs to consider the

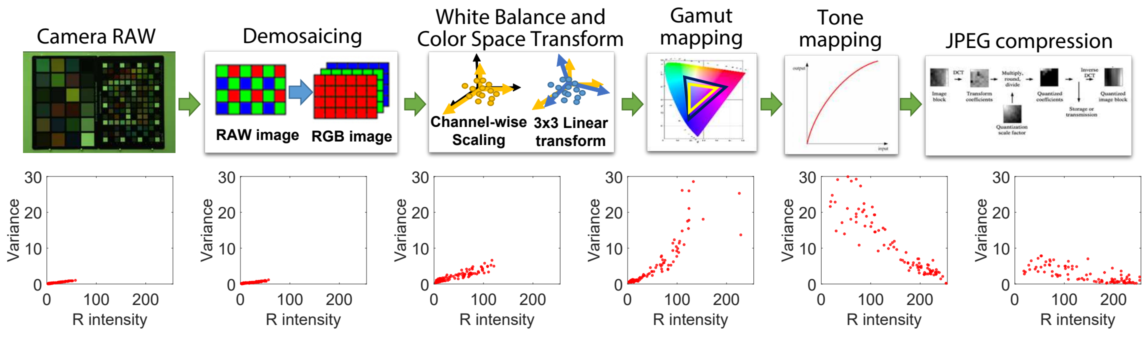


Figure 3.1: (Top) In-camera imaging pipeline. (Bottom) Changes in noise distribution through the stages of the camera imaging pipeline. Figure reproduced from [94].

correlations between noise from the three colour channels. In [94], a cross-channel noise model is characterised by a covariance matrix in the sRGB colour space

$$\mathbf{n}_i \sim \mathcal{N}\left(\mathbf{0}, \mathbf{\Sigma}(\mathbf{I}_i)\right), \quad \mathbf{\Sigma}(\mathbf{I}_i) = \begin{bmatrix} \sigma_r^2 & \sigma_{rg} & \sigma_{rb} \\ \sigma_{rg} & \sigma_g^2 & \sigma_{gb} \\ \sigma_{rb} & \sigma_{gb} & \sigma_b^2 \end{bmatrix}, \tag{3.11}$$

where $\mathbf{n}_i$ in this equation represents the three colour values (RGB) at pixel location i and $\mathcal{N}\left(\mathbf{0}, \mathbf{\Sigma}(\mathbf{I}_i)\right)$ is the zero-mean multivariate Gaussian distribution of noise with the covariance matrix $\mathbf{\Sigma}(\mathbf{I}_i)$ that is a function of the clean intensity $\mathbf{I}_i$.

Another issue that needs careful handling is the effect of colour processing on the noise distribution. Figure 3.1 shows the changes in the noise variance distribution through the camera image processing pipeline. Noise characteristics drastically change with the gamut mapping and the tone mapping processes.

Mixture models Simple parametric models, such as the examples discussed earlier, are far from capturing the real noise distributions observed from empirical data [124]. Hence, more complex models are needed to capture the complexity of real noise. A recent example of such models is the Poisson mixture model [124] where a mixture of Poisson distributions has been found to more accurately fit real noise distributions. Following the notation from Equation 3.6, the noise under a Poisson mixture model, with an additional Gaussian component for signal-independent noise, can be described as

$$n_i \sim \alpha \, z \, \mathcal{P}(\mathbf{I}_i) - \mathbf{I}_i + \mathcal{N}(0, \delta^2), \tag{3.12}$$

where z is a hidden Bernoulli variable with camera-specific constants z_1 and z_2 where

$$p[z = z_1] = 1 - p[z = z_2]. \tag{3.13}$$

The probabilities $p[z = z_1]$ and $p[z = z_2]$ are usually trained per camera model, to best match the underlying empirical noise distribution of the sensor measurements.

All of the discussed noise models in this chapter are parametric and designed under specific sets of assumptions. The homoscedastic Gaussian and Poisson models can not explicitly model the different noise components discussed in the previous chapter. The heteroscedastic models (i.e., the multivariate Poisson, the Poisson-Gaussian, and the heteroscedastic Gaussian) can explicitly model the photon noise and the thermal noise components; however, these models cannot explicitly model spatially-varying noise components such as fixed pattern noise and defective photosite noise. The models that went one step further are the cross-channel model and the clipped heteroscedastic Gaussian model that could explicitly model cross-talk noise and the clipped intensity noise. Table 3.1 shows a brief summary of the types of noise that are explicitly modelled by each of the discussed noise models. To this end, as observed from the state-of-the-art work on image noise modelling, there is still wide room for improvement in image noise modelling and closing the gap between noise models and real noise distributions.

Noise model	Noise type						
	Photon	Dark current/thermal	Intensity clipping	Cross-talk	Fixed pattern	Gain/ISO	ADC
Gaussian Assumption		✓					
Univariate Poisson		✓					
Multivariate Poisson	✓	✓					
Poisson-Gaussian	✓	✓					
Heteroscedastic Gaussian	✓	✓					
Clipped Heteroscedastic Gaussian	✓	✓	✓				
Cross-channel Gaussian	✓	✓		✓			
Poisson Mixture	✓	✓					

Table 3.1: A brief summary of the types of noise that are *explicitly* modelled be each of the discussed noise models.

Chapter 4

Image Denoising

A vast amount of research work has been conducted to address the problem of image denoising. Covering and discussing all image denoising algorithms is outside the scope of this dissertation. Excellent surveys and book chapters on this topic are available [18, 56, 15, 112, 13, 42]. Instead, this chapter will present a review of the most prominent image denoising methods from the literature followed by a discussion of image denoising strategies and common practices applied in many denoising methods. In addition, an overview of noise estimation is presented.

4.1 A Review of Image Denoising

Image denoising is the process of reducing *noise* in images, and ultimately restoring the latent clean image. Starting from the additive noise formulation, image denoising aims to solve the following under-determined equation:

$$\tilde{\mathbf{I}} = \mathbf{I} + \mathbf{n}, \tag{4.1}$$

where the given is the observed noisy image, $\tilde{\mathbf{I}}$, and it is required to separate the noisy image, $\tilde{\mathbf{I}}$, into the two parts: the latent clean image, $\mathbf{I}$, and the noise layer, $\mathbf{n}$.

The most basic noise reduction methods are based on **spatial filtering** that work under the assumption that neighbour pixels have similar values. Spatial filters reduce noise by simply aggregating the values in a neighbourhood of pixels. Examples of spatial filters used for noise reduction are box filters, Gaussian filters, and median filters. A **box filter**, also known as a box blur, is basically the arithmetic mean of a neighbourhood of pixels

$$\hat{\mathbf{I}}_{ij} = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \tilde{\mathbf{I}}_{ij}, \tag{4.2}$$

where $\hat{\mathbf{I}}_{ij}$ is the estimated clean pixel at the two-dimensional coordinates ij and M and N are the dimensions of the box filter. An example 3×3 box filter can be written as

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{4.3}$$

Box filters can be implemented very efficiently; however, they have some disadvantages, such as ringing artefacts. These artefacts happen due to the fact that box filters have zero and negative frequency components that result in removing or phase-inverting some image frequencies.

A widely used alternative to box filters is the **Gaussian filter** (i.e., Gaussian blur or Gaussian smoothing). Instead of applying arithmetic mean, a Gaussian filter uses a Gaussian function for calculating the transformation to apply to each pixel in the filter window

$$G_{mn} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{m^2 + n^2}{2\pi\sigma^2}\right), \tag{4.4}$$

where m and n are the zero-centred two-dimensional coordinates of the filter window.

An example of a 5×5 two-dimensional discrete Gaussian filter is

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}. \tag{4.5}$$

Spatial filters used for noise reduction are commonly referred to as low-pass filters as they tend to remove high-frequency components while preserving low-frequency components of the signal. This behaviour usually results in blurring the image structures, such as edges. Some spatial filters are designed to avoid blurring effects such as the median filter. The **median filter** is a rank-statistic filter that replaces each pixel with the median value from a set of neighbourhood pixels

$$\hat{\mathbf{I}}_{ij} = \underset{st \in \Omega_{ij}}{\text{median}}\{\tilde{\mathbf{I}}_{st}\}, \tag{4.6}$$

where Ω_{ij} is the set of two-dimensional coordinates of the filter's window. The median filter is also commonly used to deal with outlier noise observations, such as defective pixels.

An alternative basic approach of noise reduction is **frequency domain filtering**. Image noise can be thought of as the high frequency components of the image signal while the clean image consists of relatively lower frequency components. This division of the image signal into low and high frequency components is not always valid since image edges and textures are considered high frequency components and can be mistakenly considered as noise by the denoising algorithm. One of the basic image denoising methods that depend on the frequency division of the image signal is the **Wiener filter** [80, p. 527]. With the assumption that the noise is a zero-mean stationary random process, uncorrelated with the image signal, the Wiener filter can be used to estimate the clean image as follows

$$\hat{\mathbf{I}}_{ij} = h_{ij} * \tilde{\mathbf{I}}_{ij}, \tag{4.7}$$

where $\hat{\mathbf{I}}$ is the estimated clean image, $\tilde{\mathbf{I}}$ is the observed noisy image, h is the Wiener filter, $*$ is the convolution operator, and ij are the two-dimensional pixel coordinates in a spatial domain. The used error criterion is the mean square error (MSE)

$$e = E\left[\left(\mathbf{I}_{ij} - \hat{\mathbf{I}}_{ij}\right)^2\right], \tag{4.8}$$

where $\mathbf{I}$ is the ground-truth clean image and E is the expected or the mean value of pixel errors. In the frequency domain, the Wiener filter would be

$$\mathcal{H}_{uv} = \frac{\mathcal{G}^*_{uv}\, \mathcal{S}_{uv}(\mathbf{I})}{\left|\mathcal{G}_{uv}\right|^2 \mathcal{S}_{uv}(\mathbf{I}) + \mathcal{S}_{uv}(\mathbf{n})} \tag{4.9}$$

$$= \frac{\mathcal{G}^*_{uv}}{\left|\mathcal{G}_{uv}\right|^2 + \frac{\mathcal{S}_{uv}(\mathbf{n})}{\mathcal{S}_{uv}(\mathbf{I})}} \tag{4.10}$$

$$= \frac{\mathcal{G}^*_{uv}}{\left|\mathcal{G}_{uv}\right|^2 + \frac{1}{\mathrm{SNR}(\mathbf{I},\mathbf{n})}}, \tag{4.11}$$

where $\mathcal{H}$ is the Wiener filter, $\mathcal{G}$ is a degradation function, $\mathcal{S}(\mathbf{I})$ is the power spectrum of the clean image, $\mathcal{S}(\mathbf{n})$ is the power spectrum of the noisy image, and uv are the two-dimensional coordinates in the frequency domain. The operator SNR indicates the signal-to-noise ratio (SNR). The superscript $*$ indicates the complex conjugate. In the case of noise reduction without considering any other degradation functions, $\mathcal{G}$ becomes an identity transform and the Wiener filter becomes

$$\mathcal{H}_{uv} = \frac{1}{1 + \frac{1}{\mathrm{SNR}(\mathbf{I},\mathbf{n})}}. \tag{4.12}$$

Thus, we find that the Wiener filter operates to selectively suppress frequency components where the signal to noise ratio is smallest. Using either of Equations 4.9 or 4.12, the Wiener filtering process in the frequency domain becomes

$$\hat{\mathcal{I}}_{uv} = \mathcal{H}_{uv}\,\tilde{\mathcal{I}}_{uv}, \tag{4.13}$$

where $\tilde{\mathcal{I}}$ and $\hat{\mathcal{I}}$ are the Fourier transforms of the observed noisy image and the estimated clean image, respectively. In most practical scenarios, the power spectra of the clean image and the noise signal are unknown, and hence, the SNR is usually estimated as a scalar value instead of a power spectrum.

Using Equation 4.9, other variations of the Wiener filter can be expressed as follows

$$\mathcal{H}_{uv} = \left(\frac{\mathcal{S}_{uv}(\mathbf{I})}{\mathcal{S}_{uv}(\mathbf{I}) + \alpha_{\mathrm{w}}\mathcal{S}_{uv}(\mathbf{n})}\right)^{\beta_{\mathrm{w}}}, \tag{4.14}$$

where α_{w} and β_{w} are constants. When $\alpha_{\mathrm{w}} = 1$ and $\beta_{\mathrm{w}} = 1$, Equation 4.14 becomes the original Wiener filter. When $\alpha_{\mathrm{w}} = 1$ and $\beta_{\mathrm{w}} = 0.5$, the filter becomes a *power spectrum filter*, which is an ad hoc proxy to a human perception criterion. When α_{w} is a parameter and $\beta_{\mathrm{w}} = 1$, the filter becomes a *parametric Wiener filter*.

Another line of image denoising algorithms is based on the statistical assumption that neighbour pixels should have similar values except for the case when an edge exists. These methods are commonly referred to as **edge-preserving filtering** methods.

One of the most common edge-preserving noise filters is the **anisotropic diffusion** [98]. Anisotropic diffusion, also called *Perona-Malik diffusion*, aims at reducing image noise without removing significant edges or structures. Anisotropic diffusion is a non-linear, space-variant transformation of the original image. Anisotropic diffusion resembles the process that creates a scale space, where an image generates a parameterised family of successively blurred images based on a diffusion process. Each of the resulting images in this family is given as a convolution between the image and a 2D isotropic Gaussian filter, where the width of the filter increases with the scale parameter.

Let M_a denote the manifold of smooth images, then the diffusion equations can be interpreted as the gradient descent equations for the minimisation of the energy functional $E_a : M_a \to \mathbb{R}$ defined by

$$E_a[\tilde{\mathbf{I}}] = \frac{1}{2} \int \int \rho \left(\|\nabla \tilde{\mathbf{I}}_{ij}\|^2 \right) \, di \, dj \tag{4.15}$$

where ∇ is the gradient operator and $\rho : \mathbb{R} \to \mathbb{R}$ is a real-valued function which is intimately related to the diffusion coefficient. By solving the minimisation problem, the gradient descent equations on the functional E_a are given by

$$\frac{\partial \tilde{\mathbf{I}}}{\partial t} = -\nabla \frac{\partial E_a}{\partial \tilde{\mathbf{I}}} = \operatorname{div} \left(\rho' \left(\|\nabla \tilde{\mathbf{I}}\|^2 \right) \nabla \tilde{\mathbf{I}} \right) \tag{4.16}$$

$$= \nabla \rho' \cdot \nabla \tilde{\mathbf{I}} + \rho' \Delta \tilde{\mathbf{I}} \tag{4.17}$$

where div is the divergence operator, Δ is the Laplacian operator, and ρ' is the diffusion coefficient. In [98], two functions for the diffusion coefficient were proposed:

$$\rho' \left(\|\nabla \tilde{\mathbf{I}}\| \right) = \exp \left(- \left(\frac{\|\nabla \tilde{\mathbf{I}}\|}{K} \right)^2 \right) \tag{4.18}$$

and

$$\rho' \left(\|\nabla \tilde{\mathbf{I}}\| \right) = \frac{1}{1 + \left(\frac{\|\nabla \tilde{\mathbf{I}}\|}{K} \right)^2} \, , \tag{4.19}$$

with K a constant that controls the sensitivity to edges and is usually chosen experimentally or as a function of the noise level in the image. An alternative, less general way to represent anisotropic diffusion is simply as a combination between the original image and a set of non-linear, space-variant, locally adapted isotropic Gaussian filters, and hence, the term “anisotropic”.

Another well-known edge-preserving filter is the **bilateral filter** [117, 41]. The bilateral filter replaces the intensity of each pixel with a weighted average of intensity values from neighbour pixels. The averaging weights are based on a Gaussian distribution. Differently from a simple Gaussian filter, the averaging weights depend not only on spatial distance of pixels, but also on the intensity differences (i.e., intensity distances). The inclusion of range distances in the averaging weight results in preservation of image edges. The bilateral filtering can be defined as

$$\hat{\mathbf{I}}_{ij} = \frac{1}{W_{ij}} \sum_{kl \in \Omega_{ij}} \tilde{\mathbf{I}}_{kl}\, G_r(\left\|\mathbf{I}_{kl} - \mathbf{I}_{ij}\right\|)\, G_s(\|(k,l) - (i,j)\|), \tag{4.20}$$

where Ω_{ij} is the set of two-dimensional coordinates window centred around pixel ij, G_r is a range kernel for smoothing differences in intensities, and G_s is a spatial kernel (i.e., domain kernel) for smoothing differences in spatial coordinates. Both G_r and G_s are typically represented by Gaussian functions. W is a normalisation factor defined as

$$W_{ij} = \sum_{kl \in \Omega_{ij}} G_r(\left\|\mathbf{I}_{kl} - \mathbf{I}_{ij}\right\|)\, G_s(\|(k,l) - (i,j)\|). \tag{4.21}$$

With the assumption that the spatial and range kernel (i.e., G_s and G_r) are Gaussian kernels, the weight assigned for a pixel $\tilde{\mathbf{I}}_{kl}$ that is a neighbour of a target pixel $\tilde{\mathbf{I}}_{ij}$ is

$$w_{ijkl} \propto \exp\left(-\frac{(i-k)^2 + (j-l)^2}{2\sigma_{\mathrm{s}}^2} - \frac{\left\|\tilde{\mathbf{I}}_{ij} - \tilde{\mathbf{I}}_{kl}\right\|^2}{2\sigma_{\mathrm{r}}^2} \right), \tag{4.22}$$

where σ_{s} and σ_{r} are the Gaussian standard deviation of the spatial and range kernels, respectively, and referred to as the smoothing parameters of the bilateral filter. With careful adjustment of the smoothness parameters, the bilateral filter can be effective in edge-preserving noise reduction; however, the bilateral filter sometimes results in artefacts, such as gradient reversal (i.e., introducing virtual edges). Other variants of the bilateral filter, such as the guided filter [62], try to avoid such limitations through the use of more restrictive, yet more efficient, linear kernels.

The spatial filtering methods tend to have limited potential in noise reduction due to their dependency on the local neighbourhood of pixels. Also, increasing the size of neighbourhood (e.g., the filter size) tends to be inefficient. Another stream of noise reduction methods advanced one step beyond local neighbourhoods to searching the whole image for similar groups of neighbourhood pixels. This strategy is based on the assumption of *natural image self-similarity* where a neighbourhood of pixels (also referred to an *image patch*) can have multiple similar patches in different locations in the same image, as illustrated in Figure 4.1. This assumption has been the basis for many successful image denoising methods which are commonly referred to as **patch-based methods**.

A pioneer method that uses the self-similarity assumption is the **non-local means (NLM)** [17, 19]. The NLM algorithm tries to take advantage of the high degree of redundancy of any natural image. Such redundancy means that every small window in a natural image has many similar windows in the same image. The general form a NLM algorithm is as follows. Given a discrete noisy image $\tilde{\mathbf{I}}$, the estimated value of a pixel at 2D location ij is computed as a weighted average of all the pixels in the image

$$\mathrm{NLM}(\tilde{\mathbf{I}}_{ij}) = \sum_{\tilde{\mathbf{I}}_{kl}} w_{\mathrm{nlm}}(ij, kl)\, \tilde{\mathbf{I}}_{kl}, \tag{4.23}$$

where the weights $\{w_{\mathrm{nlm}}(ij, kl)\}_{kl}$ depend on the similarity between the pixels ij and kl and satisfy the usual conditions $0 \leq w_{\mathrm{nlm}}(ij, kl) \leq 1$ and $\sum_{kl} w_{\mathrm{nlm}}(ij, kl) = 1$. The similarity between two pixels ij and kl depends on the similarity of the pixel intensities in their corresponding neighbourhoods of pixels, a Gaussian weighted Euclidean distance can be used for this purpose [39]

$$s_{\mathrm{nlm}}(ij, kl) = \left\| \Omega_{ij}(\tilde{\mathbf{I}}) - \Omega_{kl}(\tilde{\mathbf{I}}) \right\|_2^2, \tag{4.24}$$

where Ω_{ij} and Ω_{kl} are the neighbourhood pixels (i.e., windows) surrounding pixels ij and kl, respectively. This similarity measure is adaptive to any additive white noise such that a noise alters the distance between windows in a uniform way

$$E\left(\left\|\Omega_{ij}(\tilde{\mathbf{I}}) - \Omega_{kl}(\tilde{\mathbf{I}})\right\|_2^2\right) = \left\|\Omega_{ij}(\mathbf{I}) - \Omega_{kl}(\mathbf{I})\right\|_2^2 + 2\sigma^2, \tag{4.25}$$

where $\tilde{\mathbf{I}}$ and $\mathbf{I}$ are, respectively, the noisy and clean images and σ^2 is an additive white Gaussian noise (AWGN) variance. This equality shows that, in expectation, the Euclidean distance preserves the order of similarity between pixels. So the most similar pixels to pixel ij in $\tilde{\mathbf{I}}$ also are expected to be the most similar pixels to pixel ij in $\mathbf{I}$. The weights associated with the quadratic distances are defined by

$$w_{\text{nlm}}(ij, kl) = \frac{1}{Z_{ij}} \exp\left(-\frac{s_{\text{nlm}}(ij, kl)}{h^2}\right), \tag{4.26}$$

where Z_{ij} is the normalising factor

$$Z_{ij} = \sum_{kl} \exp\left(-\frac{s_{\text{nlm}}(ij, kl)}{h^2}\right), \tag{4.27}$$

and the parameter h controls the decay of the exponential function, and therefore the decay of the weights, as a function of the Euclidean distances. The similarity windows can have different sizes and shapes to better adapt to the image. For simplicity, most realisations of the NLM algorithm use square windows of fixed size. The pixels with neighbourhood of higher similarity to the target pixel will have larger weights on average as illustrated in Figure 4.2.

Many patch-based denoising methods have been proposed based on the NLM strategy. Such methods include the block matching and 3D filtering (BM3D) [31, 32] and patch-based locally optimal Wiener (PLOW) [24]. This stream of the patch-based denoising methods depend on the assumption that the image has a locally sparse representation in some transform domain, such as the wavelet transform, or the discrete cosine transform (DCT). In such methods, the denoising process is to apply a shrinkage function to the

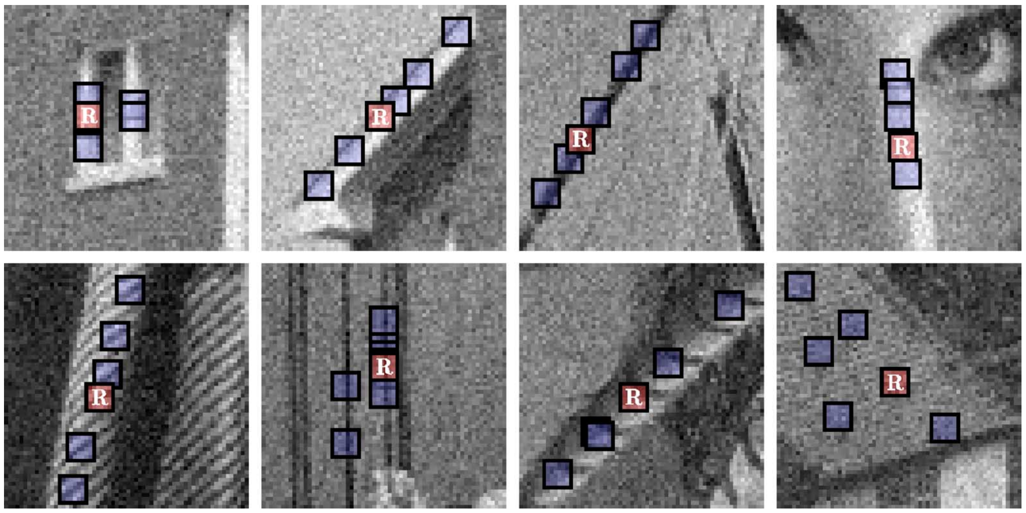


Figure 4.1: Illustration of matching blocks from noisy images corrupted by white Gaussian noise with standard deviation 15 and zero-mean. Each image shows a reference block marked with “R” and a few of the blocks matched to it. Figure reproduced from [32].

transformed coefficients and recover the denoised patches by inverting the applied transform. Patch-based methods, whether in the spatial domain or in the frequency domain, tend to be the most widely used in the literature, as illustrated in Figure 4.3. However, the performance of these patch-based algorithms highly depends on the efficiency of the underlying patch matching algorithms and how well the similar patches are matched [23]. In cases of relatively large images, patch matching algorithms tend to be highly time-consuming, and hence affecting the whole denoising process. To depart from this shortcoming of patch-based algorithms, another approach is to globally filter the image, such as the global image denoising (GLIDE) [114], where each pixel is estimated out of all pixels in the image; however, this approach is still iterative and time-consuming.

A recent wave of image denoising methods is based on deep neural networks (DNNs), especially, **convolutional neural networks (CNNs)**. A leading method in this wave is the multi-layer perceptron (MLP) denoiser [22] where the authors show that a simple, plain multi-layer perceptron can easily outperform cleverly engineered algorithms (e.g., BM3D) if trained on a large-enough dataset. In [125], it is shown that a deep CNN can outperform BM3D with the aid of batch normalisation [69] and residual learning [63].

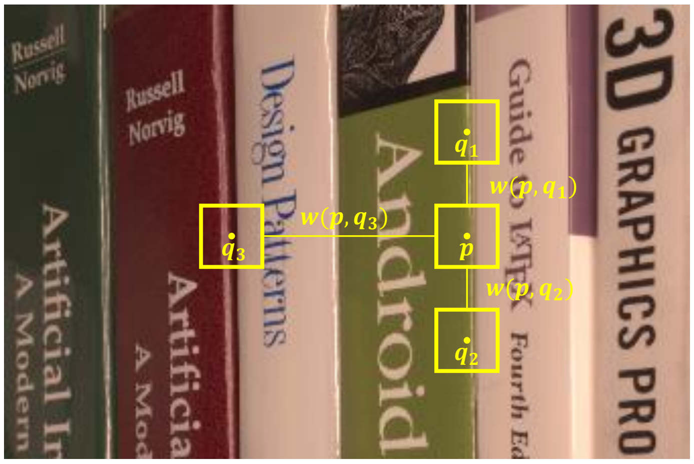


Figure 4.2: Pixels $q1$ and $q2$ have a large weight because their neighbourhood windows are similar to that of pixel p. On the other side, the weight of pixel $q3$ would be much smaller because the similarity between windows is lower.

In [113], a deep persistent memory network (MemNet) was proposed for image restoration. MemNet consists of a set of what the authors refer to as *memory blocks* that learns multi-level representations and adaptively controls how much of these representations should be reserved. In [79], the authors show that it is possible to recover signals under complex corruptions without observing clean signals and without an explicit statistical characterisation of the noise, at performance levels equal or close to using clean target data. However, such methods still require a specific noise model for generating multiple synthetic noisy observations of the same image, which may limit the performance according to the complexity of the noise model.

The aforementioned image denoising methods are among the most efficient algorithms in image denoising. On one hand, the cleverly engineered statistics-based algorithms, such as BM3D [32], are effective, but not very efficient due to the time consumed in pre- or post-processing, such as patch matching. On the other hand, learning-based methods, especially, the CNN-based ones, are very efficient, but require a great deal of training data and time. In addition, learning-based methods are commonly known for their weak generalisation to complex noise distribution, which is the case of real noisy

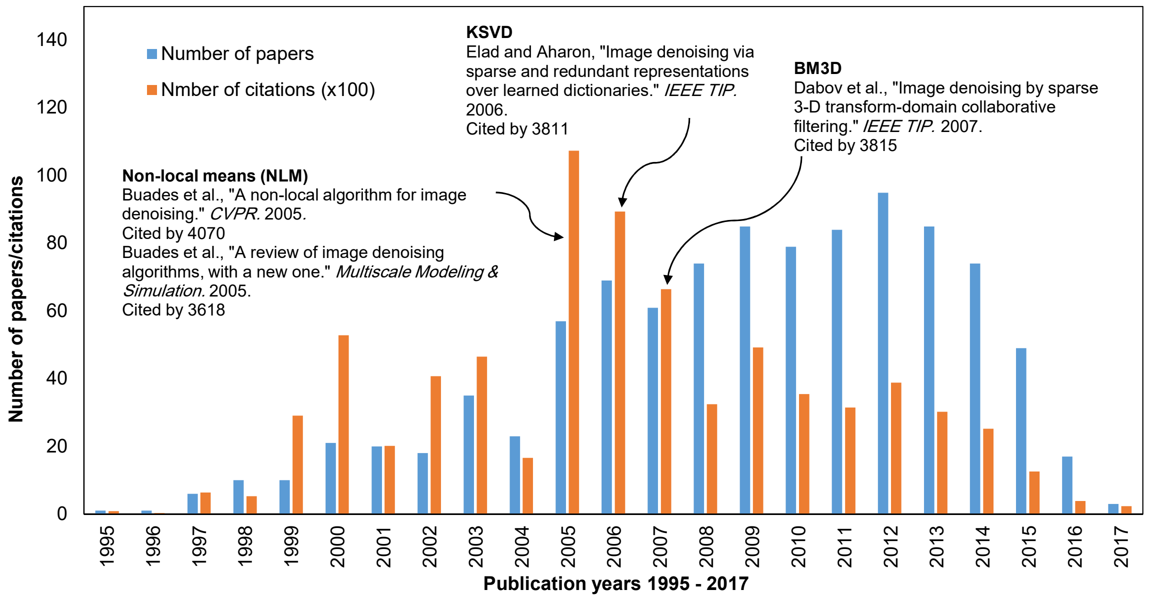


Figure 4.3: The distribution of the highly cited 980 papers with the phrase “image denoising" in the title, over the years 1995 – 2017. The number of citations of the papers from each are also shown. Data was collected from Google Scholar.

images. This is mainly due to the simplicity of the noise models (e.g., a Gaussian model) used for training such methods. Also, the evaluation of denoising methods is commonly done using synthetic noisy images as well. Some recent work has sought to combine aspects of engineered and learned systems to gain the benefits of both(e.g., [111]). Such limitations, among other common practices of image denoising research, are discussed in Sections 4.2 and 4.3.

Many computer vision algorithms require, as an input, an estimation of the noise level in an image in order to work well. This makes noise estimation an important task. However, noise estimation is tightly-coupled with the undertaken noise model as it is basically the estimation of this model’s parameters. Unlike the well-studied problem of image denoising, the literature on noise estimation is limited. Noise can be estimated from multiple images or from a single image. Both strategies are discussed next.

4.2 Multi-Image Noise Estimation

Noise estimation from multiple images is an overconstrained problem, and usually used for noise profiling of camera sensors [64, 21]. The multi-image noise estimation methods are basically based on the naive averaging of multiple images. However, many precautions need to be taken, such as minimising camera motion and lighting changes. In many situations, careful post-capture processing is required, such as defective pixel correction and spatial image registration [3]. Also, handling clipped intensities in the averaging process is needed to avoid biased averaging.

4.3 Single-Image Noise Estimation

Noise estimation from a single image, unlike the multi-image scenario, is an under-constrained problem and further assumptions need to be made. In the image denoising literature, noise is often assumed to be additive white Gaussian noise (AWGN). A widely used estimation method is based on mean absolute deviation (MAD) [38]. In [33], the authors proposed training-based methods to estimate image noise level based on training samples and the statistics of natural images.

Single noise image estimation is basically relying on the existence of homogeneous patches in the image. To avoid that, in [84, 104], the authors proposed selecting image patches with low-rank for estimating the noise. However, the performance of such methods is not theoretically guaranteed and does not have high accuracy empirically. Also, such methods tend to underestimate the noise level for the processed images, since they take the smallest eigenvalue of the covariance of selected low-rank patches as their noise estimation result [25].

In [25], the authors investigated the statistical relationship between the noise variance and the eigenvalues of the covariance matrix of patches within an image. This led to the

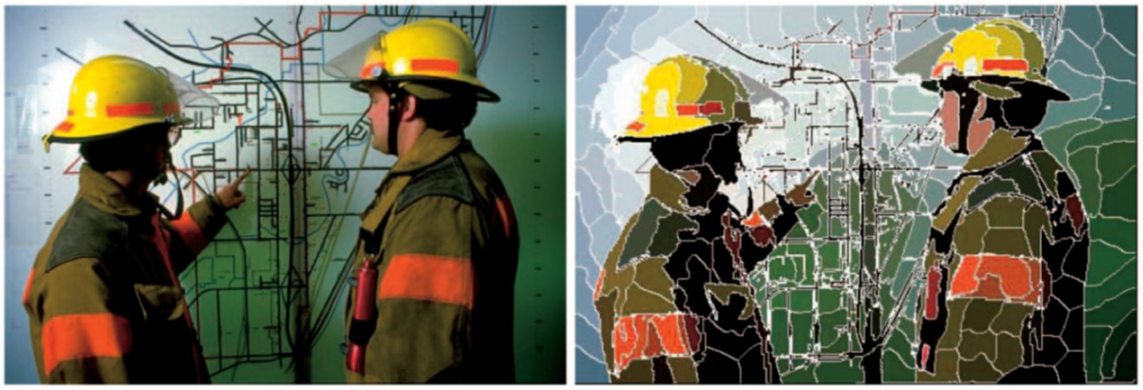


Figure 4.4: An example image illustrating the smooth image model. (Left) An input image. (Right) The image partitioned into piece-wise smooth regions using a simple K-Means clustering method for grouping pixels into regions, as described in [130]. The image is from the Berkeley image segmentation database [92]. The figure was reproduced from [82].

derivation of a non-parametric algorithm for efficient noise level estimation based on the observation that patches decomposed from a clean image often lie around a low-dimensional subspace.

All the aforementioned methods assume a signal-independent noise model and estimate a scalar standard deviation of the noise. For estimating heteroscedastic noise from a single image, a number of methods have been proposed as well. In [81, 81], piece-wise smooth image models, as illustrated in Figure 4.4, were applied to estimate noise from a single image and to help in denoising as well. In this method, the authors introduced the noise level function (NLF), which is a continuous function describing the noise level as a function of pixel intensities (see Equation 3.8). The method proposed to estimate an upper bound of the real NLF by fitting a lower envelope to the standard deviations of per-segment image variances, as illustrated in Figure 4.5. It is noted that this method is evaluated on standard RGB (sRGB) images where the cross-channel correlation of noise is more prevalent than in raw-RGB images.

Following [81, 82], a number of methods have been proposed for single-image noise estimation. In [47], the authors present a practical noise model for the raw-RGB data of

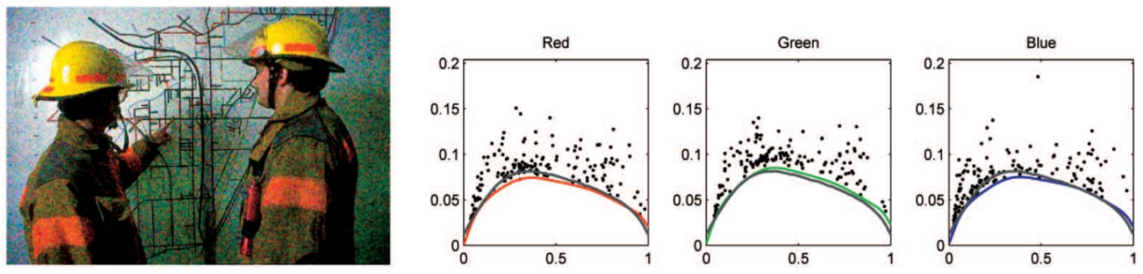


Figure 4.5: (Left) A synthesised noisy image and (right) its corresponding NLF (noise standard deviation as a function of image intensities). The red, green, and blue curves are estimated using the algorithm from [82], whereas the gray curves are the true values for the synthetically generated noise. The figure was reproduced from [82].

digital imaging sensors. They adapt the signal-dependent noise model, which gives the point-wise standard-deviation of the noise as a function of the expectation of the pixel raw intensity. Such a model is composed of a Poisson component, modelling the photon noise, and a Gaussian part, for the remaining stationary disturbances in the signal. In addition, they explicitly consider the intensity clipping effects (over- and under-exposure). As an outcome, they provide an efficient algorithm for the automatic estimation of the model parameters given a single noisy raw-RGB image.

Other work, [85], proposed an algorithm to automatically estimate the signal-dependent noise parameters from a single noisy image. The proposed algorithm identifies the noise level function of signal-dependent noise assuming the generalised signal-dependent noise model and is also applicable to the Poisson-Gaussian noise model. The main idea of this algorithm is to select the low-rank image patches, as shown in Figure 4.6; estimate the local mean and local noise variance from the selected patches; and then fit the parametric signal-dependent noise model. Despite being simple and efficient, this method was applied to sRGB images only where the signal dependency of the noise is highly impacted by the colour transformations. The method has not been evaluated on raw-RGB images.

Despite the great interest in noise estimation research, all proposed methods are targeted towards estimating the parameters of simple models, such as the Gaussian or signal-dependent models. This is to be expected as the noise estimation research usually

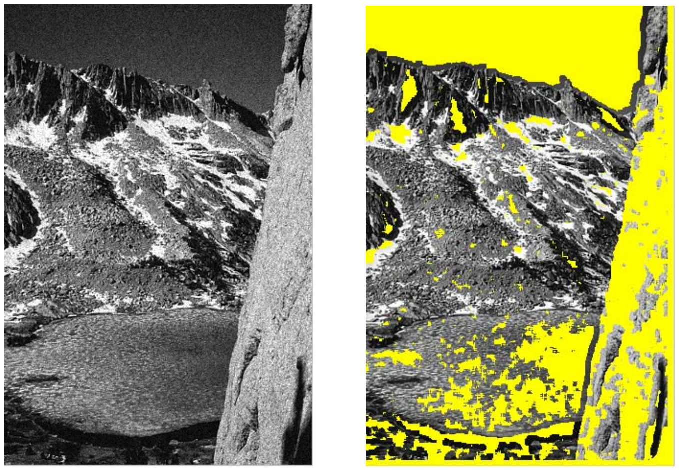


Figure 4.6: (Left) A noisy sRGB image. (Right) The selected low-rank image patches for estimating the signal-dependent noise parameters using the method from [85]. Figure reproduced from [85].

builds on already existing noise models. With both the noise model and its estimated parameters in hand, the next logical task is to perform noise reduction, i.e., image denoising. Next, we discuss a set of common practices in the image denoising field that will lay the focus on some interesting research gaps and motivate us to embark on our line of research contributions to the area of image noise modelling.

4.4 Denoising and Colour Spaces

A common practice in image denoising research is using sRGB images that are, in many cases, compressed and saved in the JPEG format, for training and evaluation of denoising methods. The colour processing of the raw-RGB images into the sRGB colour space and the possible compression into JPEG introduce dramatic changes to the noise distribution in images, as discussed in Chapter 3 and illustrated in Figure 3.1. In addition, some noise models are specific to specific colour spaces and cannot easily be generalised to all colour spaces. A clear example is the signal-dependent noise model, where the nice linear relationship between image intensity and noise variance is observed in the raw-RGB space. Once the image is converted to a non-linear colour space, this linear relationship becomes severely distorted [94].

The aforementioned practice indicates that there are some inconsistencies between the colour space of the training or evaluation images and the assumed noise model. Undertaking a signal-dependent noise model may not be the best choice for denoising tone-mapped sRGB or JPEG images. Also, assuming a Gaussian model may not best fit denoising raw-RGB images. There should be a clear consistency between the image colour space and the assumed noise model. These observations have been recently made in [94, 99, 3, 16, 5]

In addition to the proper choice of the noise model, in [99, 3], it is shown that denoising in the raw-RGB space yields higher performance rates than denoising in the sRGB space. This is due to the fact that noise in the raw-RGB follows a simpler distribution that in the sRGB space, as pointed out in [94]. However, one downside of denoising in raw-RGB space is that raw-RGB images are large ($\sim$30 MB in size), which makes them storage-consuming. The workaround to avoid saving full raw-RGB images is to efficiently compress them [95, 96, 103], or restrict the denoising process to be on-board the camera only.

4.5 Denoising and Synthetic Noise

Most denoising methods make use of synthetically generated noisy images for training and/or evaluation purposes, examples are [113, 114, 125]. The most commonly used type of synthetic noise is the additive white Gaussian noise (AWGN), assuming a Gaussian distribution. Synthetic noise is a saviour from the trouble of acquiring clean images. With synthetic noise generation, limitless amounts of training data can be produced for learning-based denoising methods. However, there are two main downsides of such a strategy. On one hand, the synthetically generated noise still requires clean images for corruption. This brings us to the same point of requiring clean images, despite a smaller number of images being needed in this case. On the other hand, synthetically generated noise tends to be far from real noise, under most of the existing noise models. This leads to degraded performances achieved on real noisy images compared to synthetically generated ones.

To this point, it is clear that the performance of a denoiser trained on synthetically generated noise is highly dependent on the complexity of the synthetic noise and how close it resembles the real noise distributions.

4.6 Denoising Datasets and Benchmarks

Image denoising research depends heavily on a number of image datasets for training and evaluation of the denoising methods. Classic examples of such datasets are the Berkeley segmentation dataset (BSD) [92], the PASCAL VOC dataset [45], the TID2008 dataset [102], and the TID2013 dataset [100, 101]. Such datasets are mostly in sRGB/JPEG space. Researchers usually use images from these datasets as the target clean images for image denoising and apply synthetic noise on them for training or evaluation.

A major issue with classic image datasets is that there is no guarantee that their images are actually noise-free and can be used directly as ground-truth for training or evaluation. In addition, the colour processing applied to such images is unknown and it is to be expected that different images may have been processed differently in terms of colour space transformation, tone mapping, photo finishing, compression, etc.

Another wave of image denoising datasets and benchmarks considered the fact that real noise is more complex than the existing noise models, and we need to deal directly with real noisy images instead of relying solely on synthetically generated ones. Recent examples in this wave are the datasets from [94, 122], the RENOIR dataset [9], the Darmstadt noise dataset (DND) [99], and the PolyU dataset [121].

There have been, to the best of our knowledge, a few attempts to quantitatively benchmark denoising algorithms on real images. One is the RENOIR dataset [9] which contains pairs of low/high-ISO images. This dataset lacks accurate spatial alignment, and the low-ISO images still contain noticeable noise. Also, the raw image intensities are linearly mapped to 8-bit depth which adversely affects the quality of the images. Another recent effort is the work on the Darmstadt noise dataset [99] (DND). Like the RENOIR dataset, DND contains pairs of low/high-ISO images. By contrast, it post-processes the low-ISO images to (1) spatially align them to their high-ISO counterparts, and (2) overcome intensity changes due to changes in ambient light or artificial light flicker. This work was the first principled attempt at producing high quality ground truth images. However, most of the images have relatively low levels of noise and normal lighting conditions, and there is a limited number of cases of high noise levels or low-light conditions, which are major concerns for image denoising and computer vision in general. Also, treating misalignment between images as a global translation is not sufficient for cases including lens motion, radial distortion, or optical image stabilisation.

To this end, the aforementioned datasets and benchmarks may not be sufficient for date-intensive experiments, such as training deep neural networks, which points at the need to have more datasets and benchmarks for real image denoising. There are several directions to fulfil these needs. One possible solution is to generate large-scale datasets of noisy images and provide a reliable method for estimating their corresponding ground truth. With such large-scale datasets in hand, reliable evaluation benchmarks can be easily provided.

The most widely used approach minimising random noise and estimating ground-truth images is *image averaging*, where the average measurement of a scene point statistically converges to the noise-free value with a sufficiently large number of images. Image averaging has become a standard technique in a broad range of imaging applications that are significantly affected by noise, including fluorescence microscopy at low light levels and astronomical imaging of dim celestial bodies. The most basic form of this approach is to capture a set of images of a static scene with a stationary camera and fixed camera settings, then directly average the images. This strategy has been employed to generate noise-free images used in evaluating a denoising method [129], in evaluating a noise estimation method [21], in comparing algorithms for estimating noise level functions [82, 85], and for determining the parameters of a cross-channel noise model [94]. While per-pixel averaging is effective in certain instances, it is not valid in two common cases: (1) when there is misalignment in the sequence of images, which leads to a blurry mean image, and (2) when there are clipped pixel intensities due to low-light conditions or over-exposure, which causes the noise to be non-zero-mean and direct averaging to be biased [46, 99]. These two cases are typical of smartphone images and are rarely discussed or addressed in the literature targeting ground-truth image estimation.

This increasing focus on estimating ground-truth for real noisy images has motivated our first research contribution to this area. In Chapter 5, we present a robust method for estimating ground truth for real noisy images. The method is mainly based on robust averaging of sequences of images. With this method, we generated a large dataset of noisy images and estimated their ground truth.

Chapter 5

A High-Quality Denoising Dataset for Smartphone Cameras

In this chapter, we present our first research contribution done towards the area of image noise modelling and estimation in general. One of the main research gaps we discussed in Chapter 4 was the lack of representative datasets and benchmarks of real noisy images with accurate ground-truth. The method we are presenting in this chapter is focused on estimating ground-truth for real noisy images. We will start by stating the motivation behind this work. Then, we will present our method and the dataset collected using this method. Finally, we will use the dataset for benchmarking a set of image denoising algorithms, revealing some interesting findings.

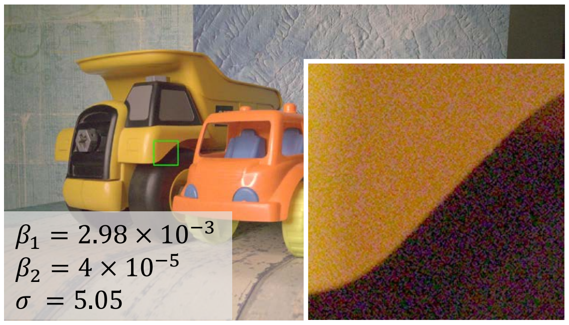
5.1 Motivation

With over 1.5 billion smartphones sold annually [50], it is unsurprising that smartphone images now vastly outnumber images captured with digital single-lens reflex (DSLR) and point-and-shoot cameras. But while the prevalence of smartphones makes them a convenient device for photography, their images are typically degraded by higher levels of noise due to the smaller sensors and lenses found in their cameras. This problem has heightened the need for progress in image denoising, particularly in the context of smartphone imagery.

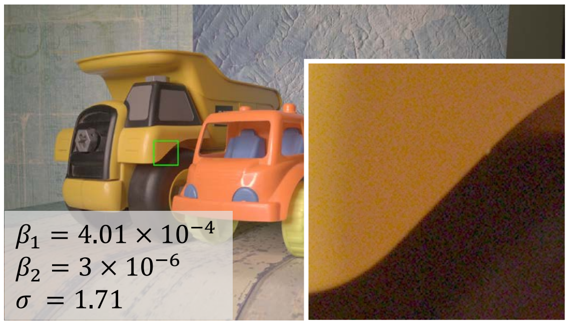
A major issue towards this end is the lack of an established benchmarking dataset for real image denoising representative of smartphone cameras. The creation of such a dataset is essential both to focus attention on denoising of smartphone images and to enable standardised evaluations of denoising techniques. However, many of the approaches used to produce noise-free ground truth images are not fully sufficient, especially for the case of smartphone cameras. For example, as discussed in Chapter 4, the common strategy of using low ISO and long exposure to acquire a "noise free" image [9, 102, 100, 101] is not applicable to smartphone cameras, as noise is still significant on such images even with the best camera settings (e.g., see Figure 5.1). Recent work in [99] moved in the right direction by globally aligning and post-processing low-ISO images to match their high-ISO counterparts. This approach performs well on DSLR cameras; however, it is not entirely applicable to smartphone images. In particular, post-processing of a low-ISO image does not sufficiently remove the remaining noise, and the reliance on a global translational alignment does not adequately align smartphone images.

In this work on ground truth image estimation, we investigate issues that are pertinent for smartphone cameras and have not been properly addressed by prior strategies, such as the effect of spatial misalignment among images due to lens motion (i.e., optical stabilisation) and radial distortion, and the effect of clipped intensities due to low-light conditions or over-exposure. In addition, we examine the impact of our dataset on recent deep learning-based methods and show that training with real noise and our ground truth leads to appreciably improved performance of such methods.

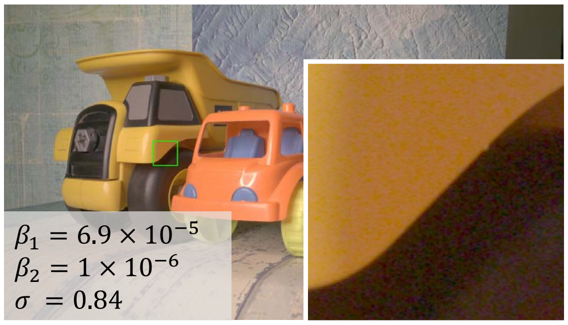
Contribution The work presented in this chapter establishes a much-needed image dataset for smartphone denoising research. To this end, we propose a systematic procedure for estimating ground truth for real noisy images that can be used to benchmark denoising performance on smartphone imagery. Using this procedure, we captured a



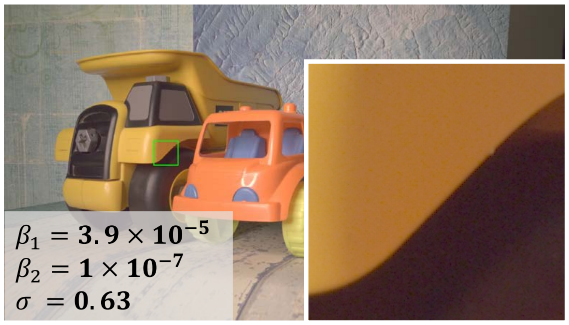
(a) Noisy image (ISO 800)



(b) Low-ISO image (ISO 100)



(c) Ground truth using [99]


$\beta_1 = \mathbf{3.9 \times 10^{-5}}$
$\beta_2 = \mathbf{1 \times 10^{-7}}$
$\sigma = \mathbf{0.63}$

(d) Our ground truth

Figure 5.1: An example of a scene imaged with an *LG G4* smartphone camera: (a) a high-ISO noisy image; (b) the same scene captured with low ISO – this type of image is often used as ground truth for (a); (c) ground truth estimated by [99]; (d) our ground truth. Noise estimates (β_1 and β_2 for noise level function and σ for Gaussian noise) indicate that our ground truth has significantly less noise than both (b) and (c). Images shown are processed in raw-RGB, while sRGB images are shown here to aid visualisation.

dataset of $\sim 30,000$ real noisy images using five representative smartphone cameras and generated their ground truth images. Using our dataset, we benchmark a number of denoising methods to gauge the relative performance of various approaches, including patch-based methods and more recent CNN-based techniques. From this analysis, we show that for CNN-based methods, notable gains can be made when using our ground truth data versus conventional alternatives such as low-ISO images.

5.2 Dataset

In this section we describe the details regarding our dataset's capture setup and protocol, and a description of the image noise estimation.

5.2.1 Image Capture Setup and Protocol

Our image capture setup is as follows. We capture static indoor scenes to avoid misalignment caused by scene motion. In addition, we use a direct current (DC) light source to avoid the flickering effect of alternating current (AC) lights [110]. Our light source allows adjustments of illumination brightness and colour temperature (ranging from 3200K to 5500K). We used five smartphone cameras (Apple iPhone 7, Google Pixel, Samsung Galaxy S6 Edge, Motorola Nexus 6, and LG G4). It is worth noting that, since all scenes are indoor, they are all man-made. This may introduce some bias in the dataset towards man-made indoor scenes; however, it would be extremely hard to apply our approach to outdoor scenes where there is no control over moving objects and lighting conditions.

We captured our dataset using the following protocol. We captured each scene multiple times using different cameras, different settings, and/or different lighting conditions, we call each combination of these conditions a *scene instance*. For each scene instance, we capture a *sequence* of successive images, with 1-2 seconds time interval between subsequent images. The 1-2 second interval is caused by the camera's image signal processor (ISP) being unable to capture RAW image fast enough. While capturing an image sequence, all camera settings (ISO, exposure, focus, white balance, exposure compensation, etc.) are fixed throughout the process.

We captured 10 different scenes, shown in Figure 5.2, using five smartphone cameras under four different combinations (on average) of the following settings and conditions:

- 15 different ISO levels ranging from 50 up to 10,000 to obtain a variety of noise levels (the higher the ISO level, the higher the noise).

Smartphone camera	Camera ID	Raw dimensions	Resolution (Mpixels)	ISO range	# scene instances	# images
Google Pixel	GP	3044×4048	12.32	$50 - 10,000$	41	$6,019$
Apple iPhone 7	IP	3024×4032	12.19	$25 - 1,600$	62	$9,300$
Samsung Galaxy S6 Edge	S6	3000×5328	15.98	$50 - 3,200$	43	$6,450$
Motorola Nexus 6	N6	3120×4208	13.13	$50 - 3,200$	30	$4,500$
LG G4	G4	2988×5312	15.87	$50 - 800$	24	$3,600$
Total					200	$29,869$

Table 5.1: Details about the smartphone cameras we used in capturing our dataset and the number of scene instances and images captured by each camera.

- Three illumination temperatures to simulate the effect of different light sources: 3200K to simulate tungsten or halogen, 4400K to simulate fluorescent lamps, and 5500K to simulate daylight.
- Three light brightness levels: low, normal, and high.

For each scene instance, we capture a sequence of 150 successive images. Therefore, the total number of images in our dataset is $\sim 30,000$ (10 scenes $\times$ 5 cameras $\times$ 4 conditions $\times$ 150 images). For each image, we record all settings together with the RAW image data in a DNG/Tiff file. Figure 5.3 shows some example images from our dataset under different lighting conditions and camera settings. In Table 5.1, we show some details about the five smartphone cameras we used to capture our dataset and the number of scene instances and the number of images captured with each camera. We show images from the twenty scene instances in our dataset captured by different cameras under different conditions in Figure 5.2. For each scene instance, we indicate the specific scene of this instance, the camera used to capture it, the number of captured images, camera settings (ISO level and exposure time), illumination temperature, and lighting conditions under which the scene instance was captured.

Throughout this chapter, we denote a sequence of images of the same scene instance

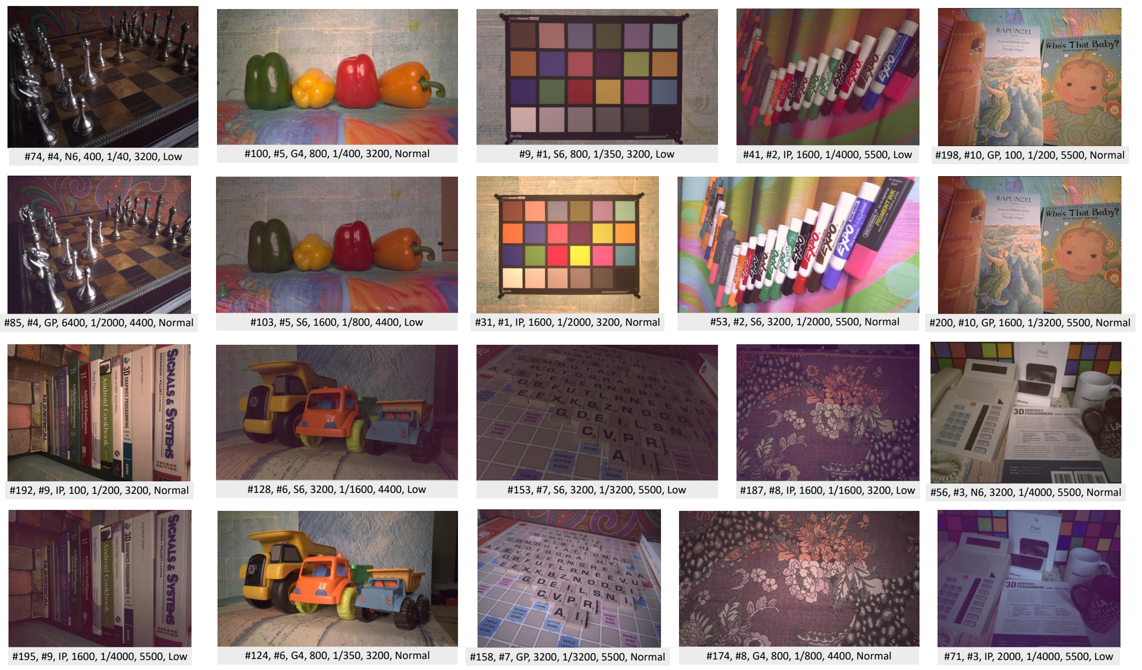

#74, #4, N6, 400, 1/40, 3200, Low
#100, #5, G4, 800, 1/400, 3200, Normal
#9, #1, S6, 800, 1/350, 3200, Low
#41, #2, IP, 1600, 1/4000, 5500, Low
#198, #10, GP, 100, 1/200, 5500, Normal
#85, #4, GP, 6400, 1/2000, 4400, Normal
#103, #5, S6, 1600, 1/800, 4400, Low
#31, #1, IP, 1600, 1/2000, 3200, Normal
#53, #2, S6, 3200, 1/2000, 5500, Normal
#200, #10, GP, 1600, 1/3200, 5500, Normal
#192, #9, IP, 100, 1/200, 3200, Normal
#128, #6, S6, 3200, 1/1600, 4400, Low
#153, #7, S6, 3200, 1/3200, 5500, Low
#187, #8, IP, 1600, 1/1600, 3200, Low
#56, #3, N6, 3200, 1/4000, 5500, Normal
#195, #9, IP, 1600, 1/4000, 5500, Low
#124, #6, G4, 800, 1/350, 3200, Normal
#158, #7, GP, 3200, 1/3200, 5500, Normal
#174, #8, G4, 800, 1/800, 4400, Normal
#71, #3, IP, 2000, 1/4000, 5500, Low

Figure 5.2: Twenty example scene instances from our dataset. The labels indicate, in the following order: the scene instance number, the scene number, the camera ID, ISO level, exposure time (seconds), illumination temperature (K), and lighting condition (low-light or normal-light). The camera IDs are indicated in Table 5.1.

as

$$\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}, \tag{5.1}$$

where $\mathbf{x}_i$ is the i^{th} image in the sequence, N is the number of images in the sequence, and $\mathbf{x}_i \in \mathbb{R}^M$, where M is the number of pixels in each image. Since we are considering images in raw-RGB space, we have only one mosaicked channel per image.

5.2.2 Noise Estimation

It is often useful to have an estimate of the noise levels present in an image. To provide such estimates for our dataset, we use two common measures. The first is the signal-dependent noise level function (NLF) [82, 47] which models the noise as a heteroscedastic Gaussian distribution where the variance of noise is proportional to the underlying image

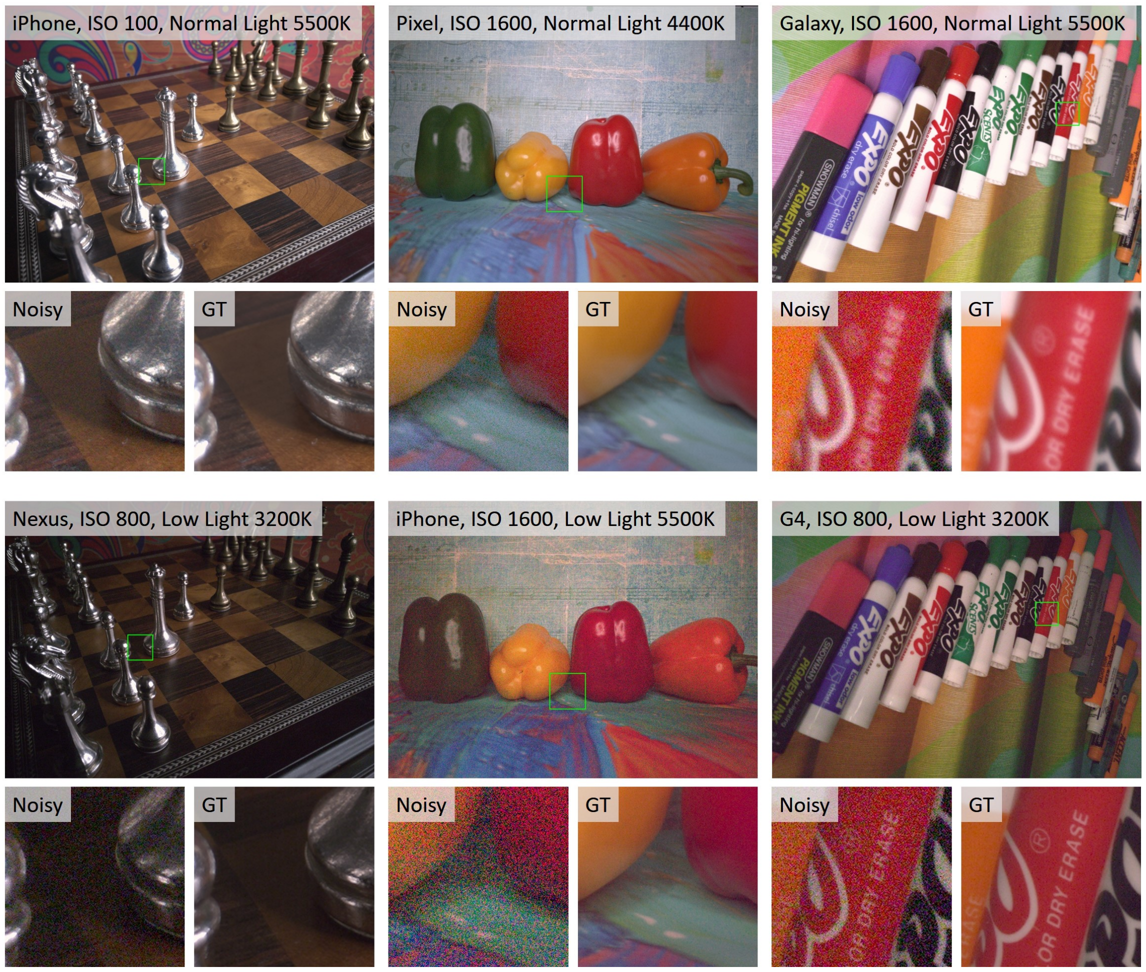

iPhone, ISO 100, Normal Light 5500K
Pixel, ISO 1600, Normal Light 4400K
Galaxy, ISO 1600, Normal Light 5500K
Noisy
GT
Noisy
GT
Noisy
GT
Nexus, ISO 800, Low Light 3200K
iPhone, ISO 1600, Low Light 5500K
G4, ISO 800, Low Light 3200K
Noisy
GT
Noisy
GT
Noisy
GT

Figure 5.3: Examples of noisy images from our dataset captured under different lighting conditions and camera settings. Below each scene, zoomed-in regions from both the noisy image and our estimated ground truth (more details in Section 5.3) are provided.

intensity. We denote the NLF-squared for a noise-free image $\mathbf{y}$ as

$$\boldsymbol{\beta}^2(\mathbf{y}) = \beta_1 \mathbf{y} + \beta_2, \tag{5.2}$$

where β_1 is the signal-dependent multiplicative component of the noise (the Poisson or shot noise), and β_2 is the independent additive Gaussian component of the noise. Then, the corresponding noisy image $\mathbf{x}$ clipped to $[0, 1]$ would be

$$\mathbf{x} = \min\Big(\max\big(\mathbf{y} + \mathcal{N}(\mathbf{0}, \boldsymbol{\beta}), \mathbf{0}\big), \mathbf{1}\Big). \tag{5.3}$$

For our noisy images, we report the NLF parameters provided by the camera device in the DNG image files, which we found to be accurate when matched with [47]. We used the Camera2 API [52] for capturing raw/DNG images on the Android phones and used a commercial application for the iPhone. To assess the quality of our ground truth images, we also measure their NLF using [47]. The second measure of noise we use is the classic homoscedastic Gaussian distribution of noise that is independent of the underlying image intensity, usually denoted by its standard deviation σ. To measure σ for our images, we use the method in [25]. We include this latter measure of noise because many denoising algorithms require it as an input parameter along with the noisy image.

5.3 Ground Truth Estimation

This section provides details on the processing pipeline for estimating ground truth images along with experimental validation of the pipeline's efficacy. Figure 5.4 provides a diagram of the major steps:

1. Capture a sequence of images following our capture setup and protocol from Section 5.2;
2. Correct defective pixels in all images (Section 5.3.1);
3. Omit outlier images and apply intensity alignment of all images in the sequence (Section 5.3.2);
4. Apply dense local image alignment of all images with respect to a single reference image (Section 5.3.3);
5. Apply robust regression to estimate the underlying true pixel intensities of the ground-truth image (Section 5.3.4).

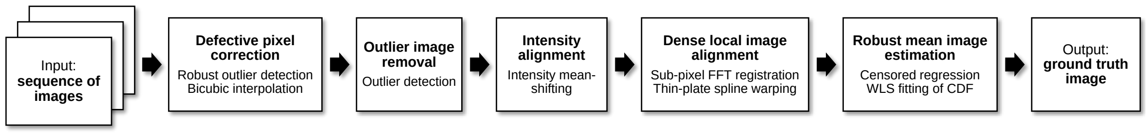

Input: sequence of images
Defective pixel correction
Robust outlier detection
Bicubic interpolation
Outlier image removal
Outlier detection
Intensity alignment
Intensity mean-shifting
Dense local image alignment
Sub-pixel FFT registration
Thin-plate spline warping
Robust mean image estimation
Censored regression
WLS fitting of CDF
Output: ground truth image

Figure 5.4: A block diagram illustrating the main steps in our procedure for ground truth image estimation.

5.3.1 Defective Pixel Correction

Defective pixels can affect the accuracy of the ground-truth estimation as they do not adhere to the same underlying random process that generates the noise at normal pixel locations. We consider two kinds of defective pixels: (1) hot pixels that produce higher signal readings than expected; and (2) stuck pixels that produce fully saturated signal readings. We avoid altering image content by applying a median filter to remove such noise and instead apply the following procedure.

First, to detect the locations of defective pixels on each camera sensor, we capture a sequence of 500 images in a light-free environment. We record the mean image denoted as $\mathbf{x}_a$, then we estimate a Gaussian distribution with mean μ_{dark} and standard deviation σ_{dark} over the distribution of pixels in the mean image $\mathbf{x}_a$. Ideally, μ_{dark} would be the *dark level* of the sensor and σ_{dark} would be the level of dark current noise. Hence, we consider all pixels having intensity values outside a 99.9% confidence interval of $\mathcal{N}(\mu_{dark}, \sigma_{dark})$ as defective pixels.

We use weighted least squares (WLS) fitting of the cumulative distribution function (CDF) to estimate the underlying Gaussian distribution of pixels. We use WLS to avoid the effect of outliers, i.e., the defective pixels, which can be up to 2% of the total pixels in the camera sensor. Also, the non-defective pixels normally have much smaller variance in their values compared to the defective pixels which leads us to use a weighted approach to robustly estimate the underlying distribution.

After detecting the defective pixel locations, we use bi-cubic interpolation to estimate the correct intensity values at such locations. Figure 5.5 shows an example of a ground

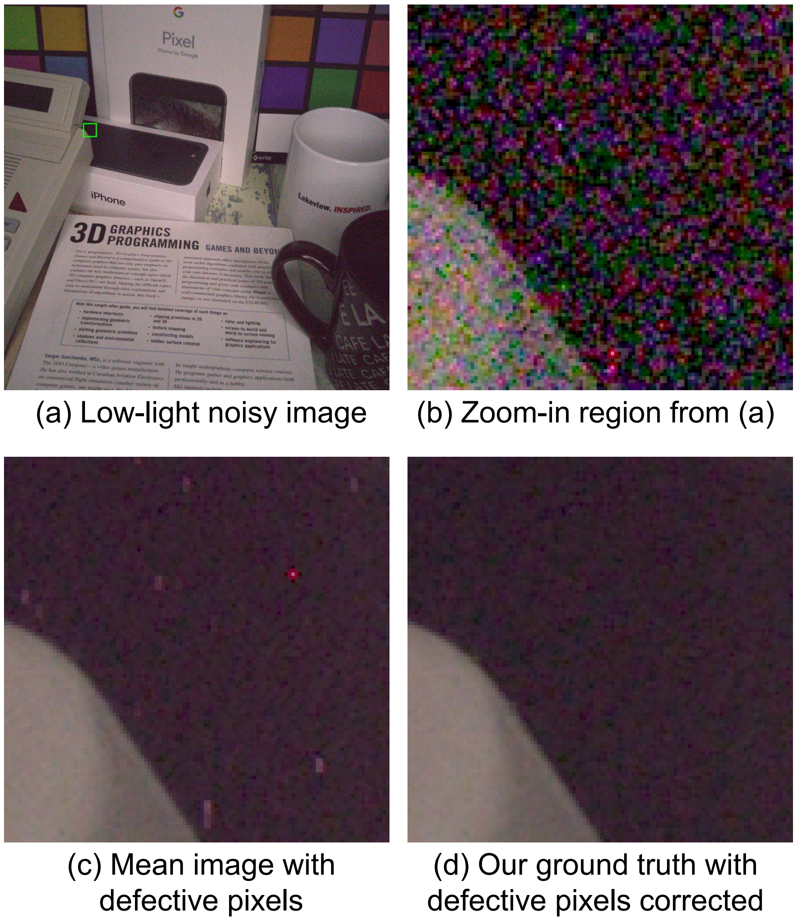
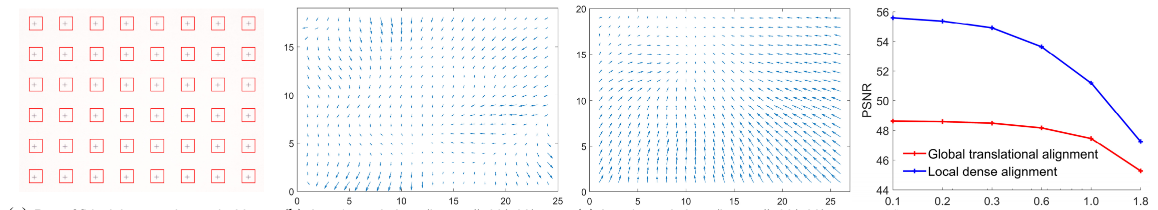

(a) Low-light noisy image
(b) Zoom-in region from (a)
(c) Mean image with defective pixels
(d) Our ground truth with defective pixels corrected

Figure 5.5: An example of a mean image (c) computed over a sequence of low-light images where defective pixels are present, and our corresponding ground truth (d) where defective pixels were corrected. One of the images from the sequence is shown in (a) and zoomed-in in (b).

truth image where we apply our defective pixel correction method versus a directly estimated mean image. In the cameras we used, the percentage of defective pixels ranged from 0.05% up to 1.86% of the total number of pixels.

5.3.2 Intensity Alignment

Despite the controlled imaging environment, there is still a need to account for slight changes in scene illumination and camera exposure time due to potential hardware imprecisions. To address this issue, we first estimate the average intensity of all images in the sequence where μ_i is the mean intensity of image i. Then, we calculate the mean μ_a



(a) Part of fiducial pattern imaged 500 times by each camera on a vibration-controlled platform.
(b) Local translations (image #500/500) Max. translation = 2.35 pixels Apple iPhone 7
(c) Local translations (image #500/500) Max. translation = 4.4 pixels Google Pixel
(d)

Figure 5.6: (a) A part of a static planar chart with fiducials imaged on a vibration-free optical table. The quiver plots of the observed and measured pixel drift between the first and last (500^{th}) image in a sequence of 500 images are shown for (b) iPhone 7 and (c) Google Pixel. (d) The effect of replacing our local alignment technique by a global 2D translation to align a sequence of images after synthesizing the local pixel drift from (b). We applied both techniques after synthesizing signal-dependent noise from a range of the β_1 parameter of the NLF estimated by the camera devices.

and standard deviation σ_a of the distribution of all μ_i and consider all images outside a 99.9% confidence interval of $\mathcal{N}(\mu_a, \sigma_a)$ as outliers and remove them from the sequence. Finally, we re-calculate μ_a and perform intensity alignment by shifting all images to have the same mean intensity:

$$\mathbf{x}_i = \mathbf{x}_i - \mu_i + \mu_a. \tag{5.4}$$

The total number of outlier images we found in our entire dataset is only 231 images. These images were typically corrupted by noticeable brightness change.

5.3.3 Dense Local Spatial Alignment

While capturing image sequences, we observed a noticeable shift in the image content over the image sequence. To examine this problem further, we placed the cameras on a vibration controlled optical table (to rule out environmental vibration) and imaged a planar scene with fixed fiducials, as shown in Figure 5.6a. We tracked these fiducials over a sequence of 500 images to reveal a spatially-varying pattern that looks like a combination of lens coaxial shift and radial distortion, as shown in Figure 5.6b for the iPhone 7 and Figure 5.6c for the Google Pixel.

On further investigation, we found that this misalignment is caused by optical image stabilization (OIS) that cannot be disabled, either through API calls, or because it was part of the underlying camera hardware [1]. As a result, we had to perform local dense alignment of all images before estimation of the ground truth image. To do this alignment, we adopted the following method for robust local alignment of the noisy images:

1. Choose one image $\mathbf{x}_{\text{ref}}$ to be a reference for the alignment of all the other images. We choose this image to be the one with the smallest deviation from the mean image $\mathbf{x}_a$:
$$\mathbf{x}_{\text{ref}} = \arg\min_{\mathbf{x}_i} \|\mathbf{x}_i - \mathbf{x}_a\|. \tag{5.5}$$

2. Divide each image into patches of size 512×512 pixels with an overlapping stride of 256 pixel along both horizontal and vertical dimensions. We choose such large patches to account for the higher noise levels in the images; the larger the patch, the more accurate our estimate of the local translation vector. We denote the centers of these patches as the destination landmarks which we use for the next registration step.

3. Use an accurate Fourier transform-based method [58] to estimate the local translation vector for each patch in each image $\mathbf{x}_i$ with respect to the corresponding patch from the reference image $\mathbf{x}_{\text{ref}}$. In this way, we obtain the source landmarks for each image.

4. Having the corresponding local translation vectors from the source landmarks in each image $\mathbf{x}_i$ to the destination landmarks in the reference image $\mathbf{x}_{\text{ref}}$, we apply 2D thin-plate spline image warping based on the set of arbitrary landmark correspondences [14] to align each image to the reference image.

We found our adopted technique to be much more accurate than treating the misalignment problem as a global 2D translation. Figure 5.6d shows the effect of replacing

[1] Google's Pixel camera does not support OIS; however, the underlying sensor, Sony's Exmor RS IMX378, includes gyro stabilization.

our local alignment technique by a global 2D translation. We applied both techniques on a sequence of synthetic images that includes synthesized local pixel shifts and signal-dependent noise. The synthesized local pixel shift is the same as measured from real images (Figure 5.6b and 5.6c). The synthesized noise is based on the noise level function (NLF) parameters (β_1 and β_2) estimated by the camera devices and extracted using the Camera2 API. Our technique for local alignment consistently yields higher PSNR values over a range of realistic noise levels versus a 2D global alignment.

In our ground truth estimation pipeline, we warp all images in a sequence to a reference image for which we desire to estimate the ground truth. To estimate ground truth for another image in the sequence, we re-apply the spatial alignment process using that image as a reference.

Verifying local misalignment of smartphone images On the observation of local pixel shifts during capture of image sequences using smartphone cameras, we conducted several experiments to estimate this local shift. We prepared a planar chart with easy-to-track fiducials as shown in Figure 5.7. We captured sequences of 500 images of this chart using all cameras. In Figure 5.7, we show the estimated local pixel shift over the 500 images for three cameras: Google Pixel, Motorola Nexus 6, and Apple iPhone. The plots indicate that the local pixel shift is unique, and the pattern is different per camera. Also, the magnitude of shift varies significantly among cameras.

In Figure 5.8, we show quiver plots of the local pixel shift between the 1st and 500th frame of each of the three image sequences. It is clear that the pixel shift is spatially varying, and a global translation cannot adequately align such images. Hence, we use 2D thin-plate spline interpolation to estimate the local per-pixel translation vectors. In our ground truth estimation pipeline, we use the same technique to locally align image sequences. However, instead of having fiducials, we have to estimate the source and destination landmarks for the thin-plate spline interpolation based on the content of images.

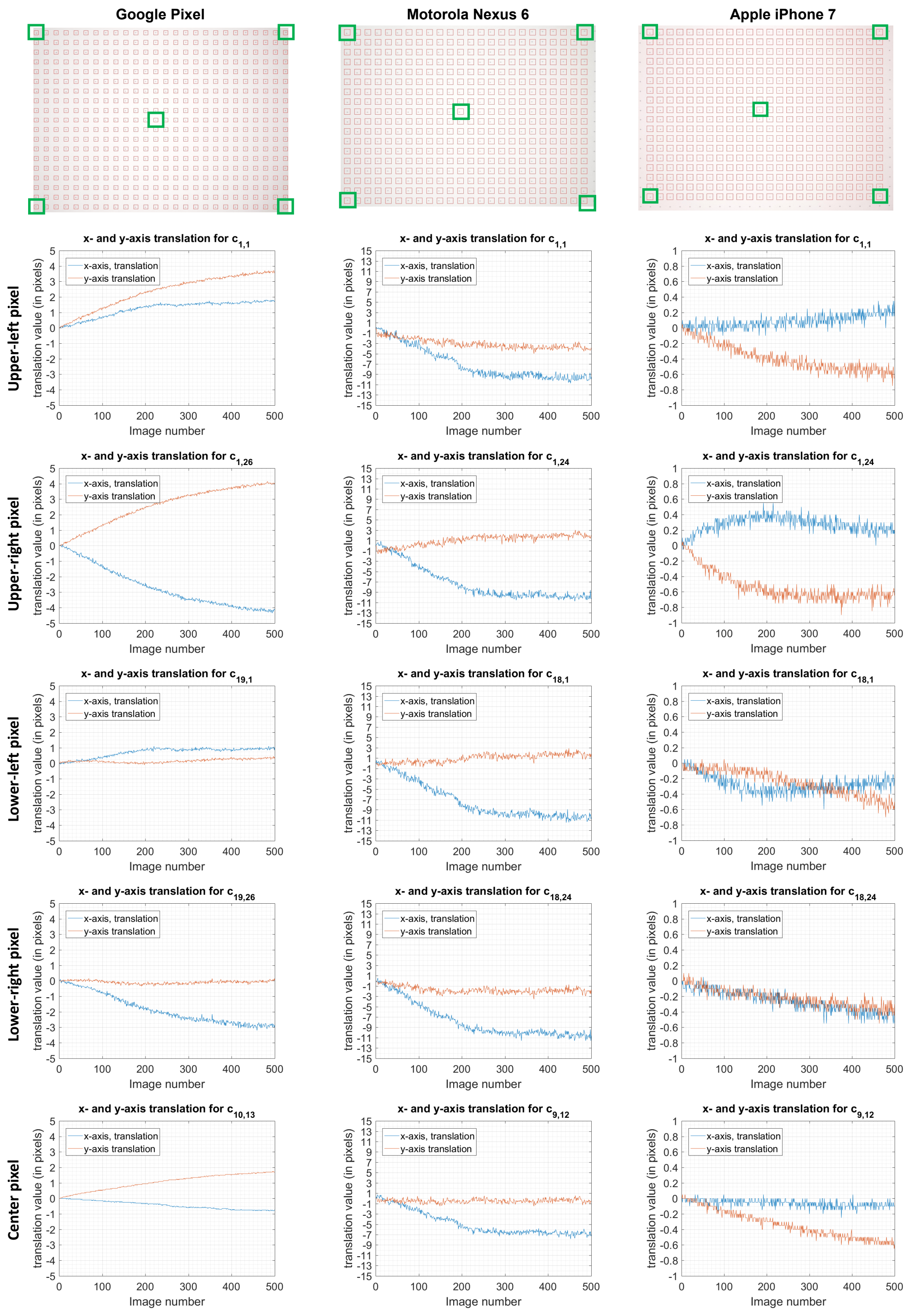


Figure 5.7: Local pixel shifts throughout a sequence of 500 images as indicated from a planar chart of fiducials. We plot the pixel shifts along the x-axis and y-axis at the five locations marked by green squares. We show results for Google Pixel, Motorola Nexus 6, and Apple iPhone 7.

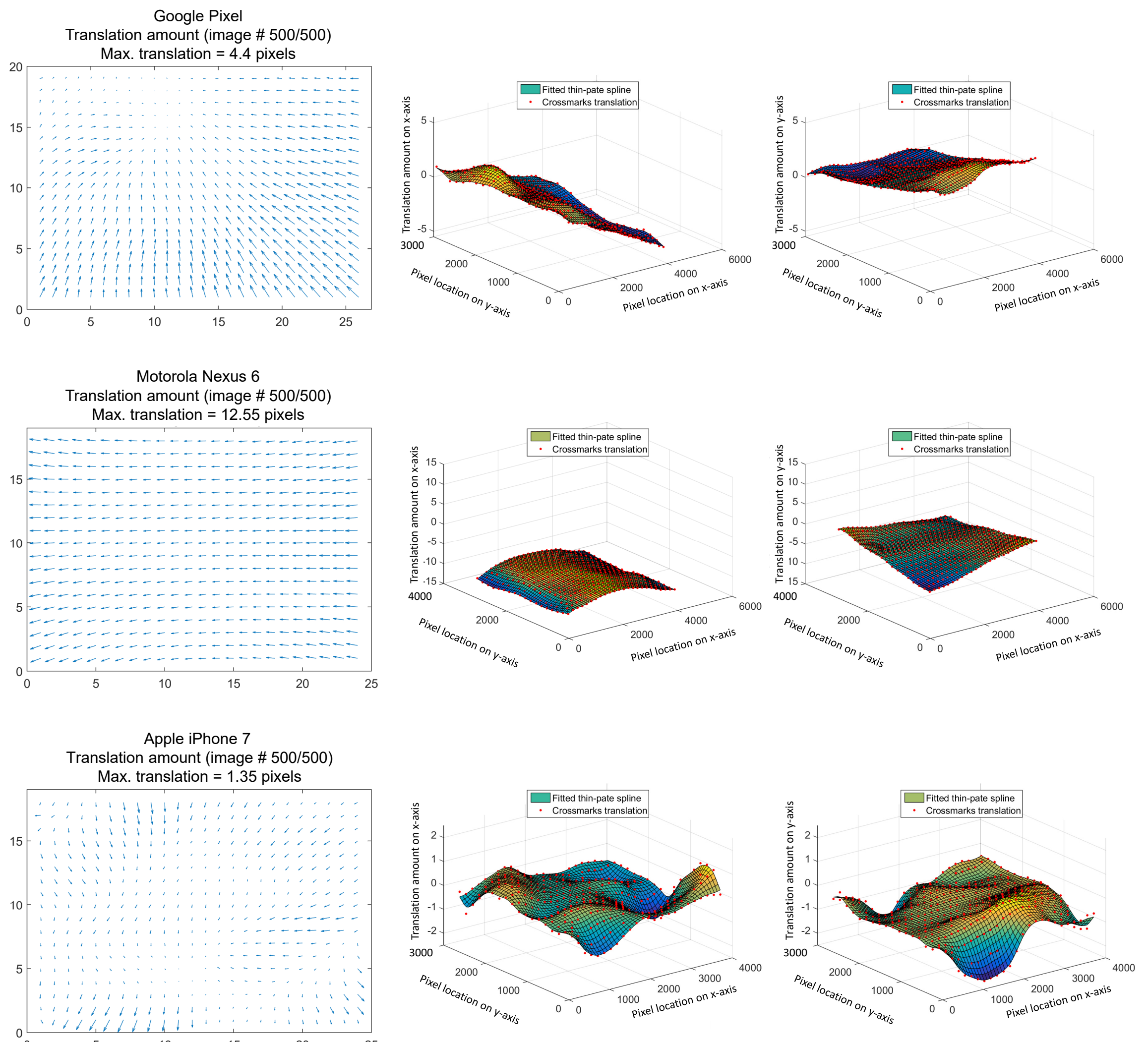


Figure 5.8: The first column shows the local pixel shift between the 1st and the 500th frames of an image sequence of the fiducials chart shown in Figure 5.7. The second and third columns show the interpolation of local pixel shifts using 2D thin-plate splines. We show results for Google Pixel, Motorola Nexus 6, and Apple iPhone 7.

5.3.4 Robust Mean Image Estimation

Once images are aligned, the next step is to estimate the mean image. The direct mean will be biased due to the clipping effects of the under-illuminated or over-exposed pixels [46]. To address this issue, we propose a robust technique that accounts for such clipping effects. Considering all observations of a pixel at position j throughout a sequence of N images, denoted as

$$\chi_j = \{x_{1j}, \ldots, x_{Nj}\}, \tag{5.6}$$

we need to robustly estimate the underlying noise-free value $\hat{\mu}_j$ of this pixel with the existence of censored observations due to the sensor's minimum and maximum measurement limits. As a result, instead of simple calculation of the mean value of χ_j, we apply the following method for robust estimation of $\hat{\mu}_j$:

1. Remove the possibly-censored observations whose intensities are equal to 0 or 1 in normalized linear RAW space:

   $$\chi'_j \leftarrow \{x_{ij} \mid \; x_{ij} \in (0, 1)\}_{i=1}^{N}, \tag{5.7}$$

   where $|\chi_j|$ becomes $N' \leq N$.

2. Define the empirical cumulative distribution function (CDF) of χ'_j as

   $$\mathbf{\Phi}_e(t \mid \chi'_j) = \sum_{i=1}^{N'} \{x_{ij} \mid x_{ij} \leq t\} / \sum_{i=1}^{N'} x_{ij}. \tag{5.8}$$

3. Define a parametric cumulative distribution function of a normal distribution with mean μ_p and standard deviation σ_p as

   $$\mathbf{\Phi}_p(t \mid \mu_p, \sigma_p) = \int_{-\infty}^{t} \mathcal{N}(t' \mid \mu_p, \sigma_p) \, dt'. \tag{5.9}$$

4. Define an objective function that represents a weighted sum of square errors between $\mathbf{\Phi}_e$ and $\mathbf{\Phi}_p$ as

$$\psi(\mu_p, \sigma_p) = \sum_{t \in \chi'_j} w_t \left(\mathbf{\Phi}_p(t \mid \mu_p, \sigma_p) - \mathbf{\Phi}_e(t \mid \chi'_j) \right)^2, \tag{5.10}$$

where we choose the weights w_t to represent a convex function such that the weights compensate for the variances of the fitted CDF values, which are lowest near the mean and highest in the tails of the distribution:

$$w_t = \left(\mathbf{\Phi}_e(t \mid \chi'_j) \left(1 - \mathbf{\Phi}_e(t \mid \chi'_j)\right) \right)^{-\frac{1}{2}}. \tag{5.11}$$

5. Estimate the mean $\hat{\mu}_j$ and standard deviation $\hat{\sigma}_j$ of χ'_j by minimizing Equation 5.10:

$$(\hat{\mu}_j, \hat{\sigma}_j) = \underset{\mu_p, \sigma_p}{\arg\min}\, \psi(\mu_p, \sigma_p) \tag{5.12}$$

using a derivative-free simplex search method [78].

To evaluate our adopted WLS method for estimating mean images affected by intensity clipping, we conduct an experiment on synthetic images with synthetic noise added and intensity clipping applied. We used NLF parameters estimated from real images to synthesize the noise. We then apply our method to estimate the mean image. We compared the result with maximum likelihood estimation (MLE) with censoring, which is commonly used for censored data regression, as shown in Figure 5.9. We repeated the experiment over a range of numbers of images (Figure 5.9a) and a range of synthetic NLFs (Figure 5.9b). For reference, we plot the error of the direct calculation of the mean image before (green line) and after (black line) applying the intensity clipping. Our adopted WLS method achieves much lower error than MLE, almost as low as the direct calculation of the mean image before clipping.

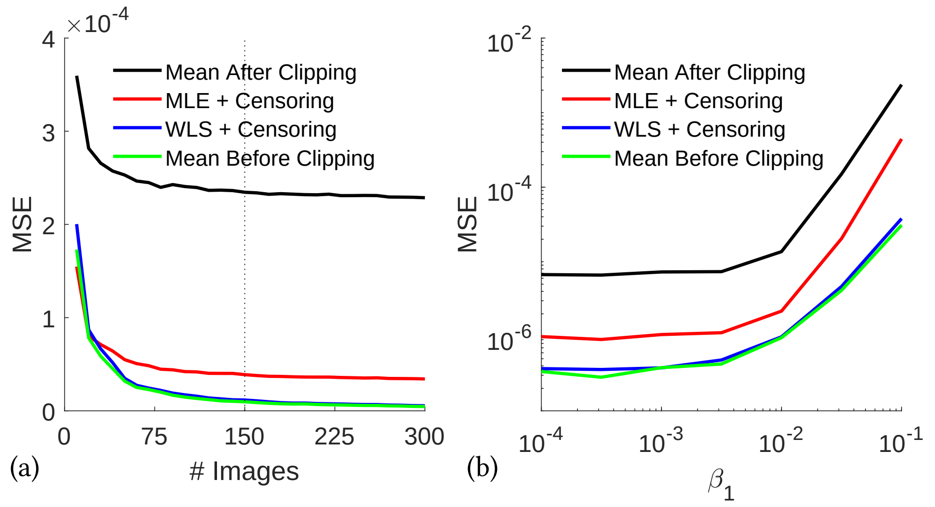


Figure 5.9: Comparison between methods used for estimating the mean image (a) over a range of number of images and (b) over a range of the first parameter of signal-dependent noise (β_1). The adopted method, WLS fitting of the CDF with censoring, yields the lowest MSE.

5.3.5 Quality of our ground truth versus the DND dataset

In order to assess the quality of ground truth images estimated by our pipeline compared to the DND post-processing [99], we asked the authors of DND to post-process five of our low/high-ISO image pairs. We then estimated the inherent noise levels in these images using [25] and compared them to our ground truth of the same five scenes as shown in Figure 5.10a. Our pipeline yields lower noise levels, hence, higher quality images, in four out of five images. Also, Figure 5.10b shows the distribution of noise levels in our dataset compared to the DND dataset. The wider range of noise levels in our dataset makes it a more comprehensive benchmark for testing on different imaging conditions and more representative for smartphone camera images. Despite having lower noise levels than our dataset, the DND has some peculiarities better than our dataset where it contains a number of outdoor scenes that represent natural objects and natural lighting conditions.

Ground truth estimation from low- and high-ISO image paris The estimation of ground-truth images from pairs of low- and high-ISO images involves many steps, as

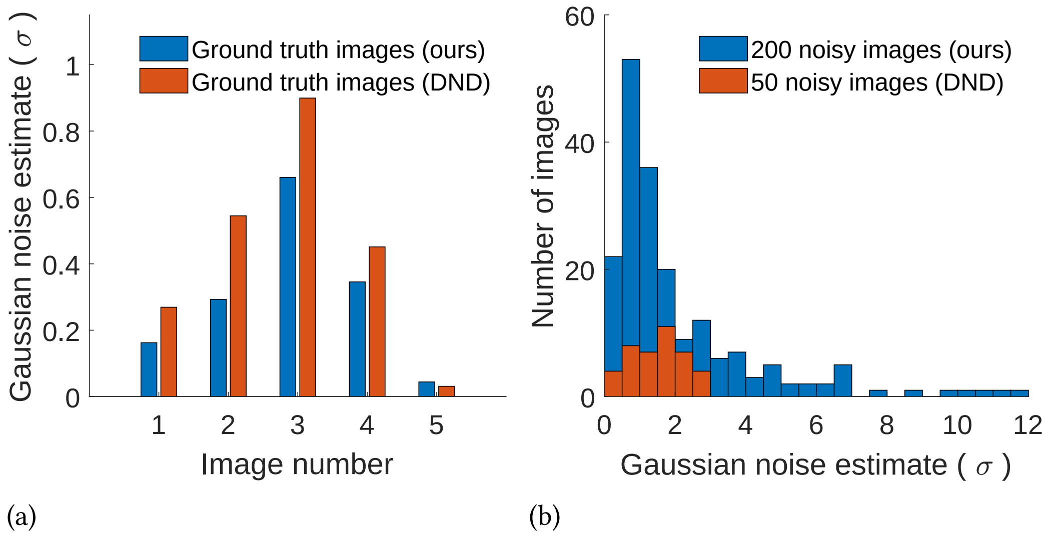


Figure 5.10: (a) Comparison between noise levels in our ground truth images versus the ground truth estimated by [99] for five scenes. Our ground truth has lower noise levels in four out of five images. (b) Comparison of noise levels in our dataset versus DND dataset.

described in [99]. First, we need to manually masking in moving objects between the two image. Then, we apply a linear intensity transfer function so that both images have matching intensities. Then, we apply global spatial alignment by registering the low-ISO image to the high-ISO one. Finally, low-frequency differences between the two images is removed using large-support smoothing filers.

Our Implementation of the DND Method In our implementation of the DND method, we applied our local alignment method instead of a global translation used in DND. We chose to apply local alignment in order to handle the local pixel shift present on smartphone images which may not exist on DSLR images. We have verified that our local alignment method is more accurate than global translation as shown in Figure 5.6. Furthermore, we used our own robust mean estimation to align the intensities of the low/high-ISO image pairs. We also verified our method's high accuracy in handling intensity clipping effects as shown in Figure 5.6. Since all our scenes are static and the camera is stationary, there was no need to apply any manual masking of moving objects. Also, we did not apply low-frequency removal since we used a DC light as the only light

	Applied/ Evaluated	BM3D [32]	NLM [20]	KSVD [7]	KSVD-DCT [43]	KSVD-G [43]	LPG-PCA [127]	FoE [109]	MLP [22]	WN-NM [57]	GLIDE [114]	TNRD [28]	EPLL [131]	Dn-CNN [125]
PSNR	Raw/Raw	**45.10**	44.06	43.26	42.70	42.50	42.79	43.13	43.17	44.85	41.87	42.77	40.73	43.30
	Raw/sRGB	**30.95**	29.39	27.41	28.21	28.13	30.01	27.18	27.52	29.54	25.98	26.99	25.19	28.24
	sRGB/sRGB	25.65	26.75	26.88	**27.51**	27.19	24.49	25.58	24.71	25.78	24.71	24.73	27.11	23.66
SSIM	Raw/Raw	**0.980**	0.971	0.969	0.970	0.969	0.974	0.969	0.965	0.975	0.949	0.945	0.935	0.965
	Raw/sRGB	0.863	0.846	0.832	0.784	0.781	0.854	0.812	0.788	**0.888**	0.816	0.744	0.842	0.829
	sRGB/sRGB	0.685	0.699	0.842	0.780	0.771	0.681	0.792	0.641	0.809	0.774	0.643	**0.870**	0.583
Time	Raw	16.9	210.7	2243.9	133.3	153.6	438.1	6097.2	131.2	1975.8	12440.	**15.2**	653.1	51.7
	sRGB	**27.4**	621.9	9881.0	96.3	92.2	2004.3	12166.8	564.8	8882.2	36091.	45.1	1996.4	158.9

Table 5.2: Denoising performance in terms of PSNR (dB), SSIM, and denoising time in seconds per 1-Mpixel image (1024×1024 pixels) for the benchmarked methods averaged over 40 images. Top three methods are indicated with colours (green, blue, and red) in top-down order of performance, with best results in bold. For reference, the mean PSNRs of benchmark images in RAW and sRGB are 36.70 dB and 19.71 dB, respectively, and the mean SSIM values are 0.832 and 0.397 in RAW and sRGB, respectively. It is worth noting that the mean PSNRs of the noisy images in [99] were reported as 39.39 dB (RAW) and 29.98 (sRGB), which indicate lower noise levels than in our dataset.

source and the low-frequency patterns mainly occur due to AC light flicker.

5.4 Benchmark

In this section we benchmark a number of representative and state-of-the-art denoising algorithms to examine their performance on real noisy images with our recovered ground truth. We also show that the performance of CNN-based denoising algorithms can be significantly improved if trained on real noisy images with our ground truth instead of synthetically generated noisy images and/or low-ISO images as ground truth.

5.4.1 Setup

For the purpose of benchmarking, we picked 200 ground truth images, one for each scene instance in our dataset. From these 200 images, we carefully selected a representative subset of 40 images for evaluation experiments and for a benchmarking website to be released as well, while the other 160 noisy images and their ground truth images will be made available for training purposes. Since many of the denoisers are computationally expensive (some taking more than one hour to denoise a 1-Mpixel image), we expedite comparison by applying all of the denoisers on 32 randomly selected non-overlapping image patches of size 256×256 pixels from each of the 40 images, for a total of 1280 image patches. The computation times of the benchmarked algorithms were obtained by running all of them single-threaded on the same machine equipped with an Intel® Xeon® CPU E5-2637 v4 @ 3.50GHz with 128GB of memory.

The algorithms we chose to benchmark are: BM3D [32], NLM [20], KSVD [7], LPG-PCA [127], FoE [109], MLP [22], WNNM [57], GLIDE [114], TNRD [28], EPLL [131], and DnCNN [125]. For KSVD, we benchmark two variants of the original algorithm [43], one using the discrete cosine transform (DCT) over-complete dictionary, denoted here as KSVD-DCT, and the other using a global dictionary of natural image patches, denoted here as KSVD-G. For benchmarking the learning-based algorithms (e.g., MLP, TNRD, and DnCNN), we use the available trained models for the sake of fair comparison against other algorithms; however, in Section 5.4.3 we show the advantage of training DnCNN on our dataset. We applied all algorithms in both raw-RGB and sRGB spaces. However, the denoising in raw-RGB space is evaluated in both RAW space and after conversion to sRGB space. In all cases, we evaluate the performance against our ground truth images. To render images from RAW to sRGB, we use the camera platform from [71] to simulate the camera processing pipeline using metadata from the DNG files.

Most of the benchmarked algorithms require, as input parameter, an estimate of the noise present in the image in the form of either the standard deviation of a uniform-power Gaussian distribution (σ) or the two parameters (β_1 and β_2) of the signal-dependent noise level function. We follow the same procedure from Section 5.2.2 to provide such estimates of the noise as input to the algorithms.

5.4.2 Results and Discussion

Table 5.2 shows the performance of the benchmarked denoising methods in terms of peak signal-to-noise ratio (PSNR), structural similarity (SSIM) [118], and denoising time. Our discussion; however, will be focused on the PSNR-based ranking of methods, as the top-performing methods tend to have similar SSIM scores, especially in RAW space. From the PSNR results, we can see that classic patch-based and optimization-based methods (e.g., BM3D, KSVD, LPG-PCA, and WNNM) are outperforming learning-based methods (e.g., MLP, TNRD, and DnCNN) when tested on real images. This finding was also observed in [99]. We additionally benchmarked a number of methods not examined in [99] and make some interesting observations. One is that the two variants of the classic KSVD algorithm, trained on DCT and global dictionaries, achieve the best and second best PSNRs for the case of denoising in sRGB space. This is mainly because the underlying dictionaries well-represent the distribution of small image patches in the sRGB space. Another observation is that denoising in the RAW space yields higher quality with faster denoising compared to denoising in the sRGB space, as shown in Table 5.2. Also, we can see that BM3D is still one of the fastest denoising algorithms in the literature along with TNRD and dictionary-based KSVD, followed by other discriminative methods (e.g., DnCNN and MLP) and NLM. Furthermore, this examination of denoising times raises concerns about the applicability of some denoising methods. For example, though WNNM is one of the best denoisers, it is also among the slowest. Overall, we find the BM3D algorithm to remain one of the best performers in terms of denoising quality and computation time combined.

5.4.3 Application to CNN Training

To further investigate the usefulness of our high-quality ground truth images, we use them to train the DnCNN denoising model [125] and compare the results with the same

	# patches	# training	# testing	$[\sigma_{\min}, \sigma_{\max}]$	σ_μ
Subset A	5,120	4,096	1,024	[1.62, 5.26]	2.62
Subset B	10,240	8,192	2,048	[4.79, 23.5]	9.73

Table 5.3: Details on the two subsets of RAW image patches we used for training the DnCNN denoising model. $\sigma_{\min}$, $\sigma_{\max}$, and σ_μ indicate the minimum, maximum, and mean noise levels in the subsets.

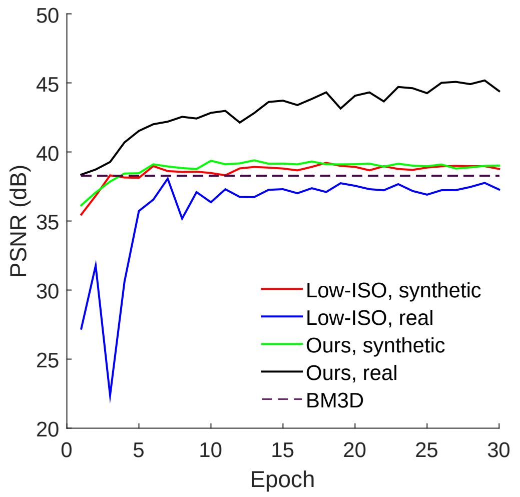
		Low-ISO		Ours	
		Synthetic	Real	Synthetic	Real
Subset A	β_1	4.66×10^{-3}	2.75×10^{-3}	2.88×10^{-3}	$\mathbf{1.01 \times 10^{-3}}$
	β_2	$\mathbf{1.90 \times 10^{-4}}$	3.94×10^{-4}	6.26×10^{-4}	8.05×10^{-4}
	σ	1.24	8.02×10^{-1}	8.95×10^{-1}	$\mathbf{4.62 \times 10^{-1}}$
Subset B	β_1	3.06×10^{-3}	2.20×10^{-3}	2.42×10^{-3}	$\mathbf{1.05 \times 10^{-3}}$
	β_2	9.67×10^{-4}	1.88×10^{-3}	$\mathbf{3.18 \times 10^{-4}}$	5.96×10^{-4}
	σ	1.03	1.04	6.97×10^{-1}	$\mathbf{4.18 \times 10^{-1}}$

Table 5.4: Mean noise estimates (β_1, β_2, and σ) of the denoised testing image patches using the four DnCNN models trained on subsets A and B. Training on our ground truth with real noise mostly yields higher quality images.

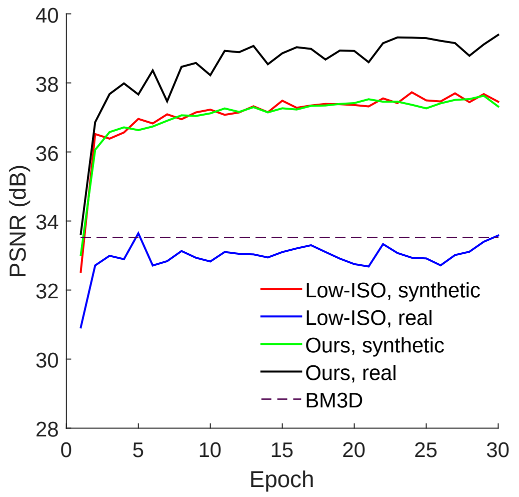
model trained on post-processed low-ISO images [99] as another type of ground truth. For each type of ground truth, we train DnCNN in two ways: one way using the real noisy images as input; the other way using the ground truth images with synthetic Gaussian noise added. For the synthetic noise, we use the mean noise level (σ_μ), as estimated from the real noisy image, to synthesize the noise. We found that using noise levels higher than σ_μ for training yields lower testing performance. To further assess the four training cases, we test on two subsets of randomly selected RAW image patches, one with low noise levels, and the other having medium to high noise levels. More details about the two subsets are shown in Table 5.3. Since we had access to only five low-ISO images post-processed by [99], we used them in subset A, whereas for subset B, we had to post-process additional low-ISO images using our own implementation of [99]. In all four cases of training, we test the performance against our ground truth images.

Figure 5.11 shows the testing results of DnCNN using two types of ground truth for training (post-processed low-ISO versus our ground truth images) and two types of noise (synthetic and real). Results are shown for both subsets A and B. We can see that training on our ground truth using real noise yields the highest PSNRs. Whereas using low-ISO ground truth with real noise yields lower PSNRs. One reason for this is the remaining noise in the low-ISO images. Also, the post-processing may not sufficiently undo the intensity and spatial misalignment between low- and high-ISO images. Furthermore, we can see that the models trained on synthetic noise perform similarly regardless of the underlying ground truth. This behaviour is likely due to both models being trained on the same Gaussian distribution of noise and therefore learn to model this same distribution. Additionally, we notice that BM3D performs comparably on low noise levels (subset A), while DnCNN trained on our ground truth images significantly outperforms BM3D on all noise levels (both subsets). To investigate if there is a bias for using our ground truth as the reference of evaluation, we compare the no-reference noise estimates (β_1, β_2, and σ) from the denoised image patches from the four models. As shown in Table 5.4, training on our ground truth with real noise mostly yields the highest quality, especially for β_1 which is the dominant component of the signal-dependent noise.

Training DnCNN on Various Noise Levels Training DnCNN [125] on synthetic noise generated using the estimated average noise level (σ_μ) of the testing images yields better results than using higher or lower noise levels. In Figure 5.12, we show the training and testing of DnCNN on image subsets A and B, as shown in Table 5.3, using a range of noise levels ($\sigma = 5, 10, 15, 20, 25$). Despite some noise levels (e.g., $\sigma = 15$ for subset A and $\sigma = 5$ for subset B) showing lower training/validation errors than for σ_μ, the best performance is still achieved by training on $\sigma_\mu = 2.62$ for subset A and $\sigma_\mu = 9.73$ for subset B, as shown in the testing plots in Figure 5.12.



(a) Subset A

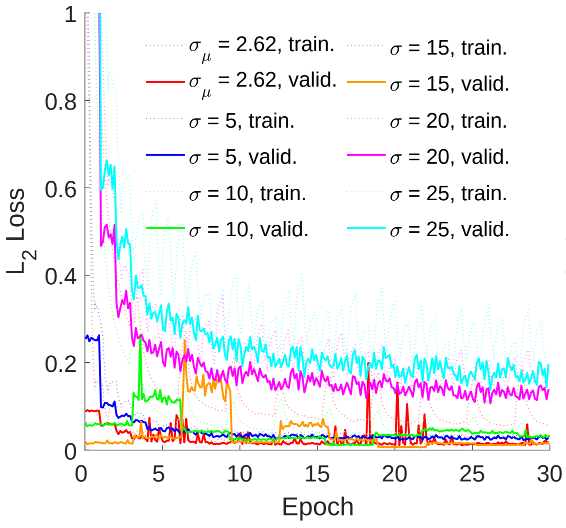


(b) Subset B

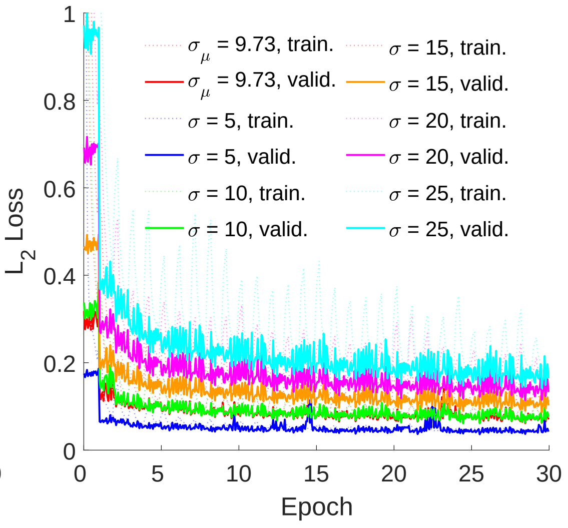
Figure 5.11: Testing results of DnCNN [125] using two types of ground truth (post-processed low-ISO images and our ground truth images) and two types of noise (synthetic and real) on two randomly selected subsets of our dataset (shown in Table 5.3). Training using our ground truth images on real noise yields the highest PSNRs.

5.5 Conclusion

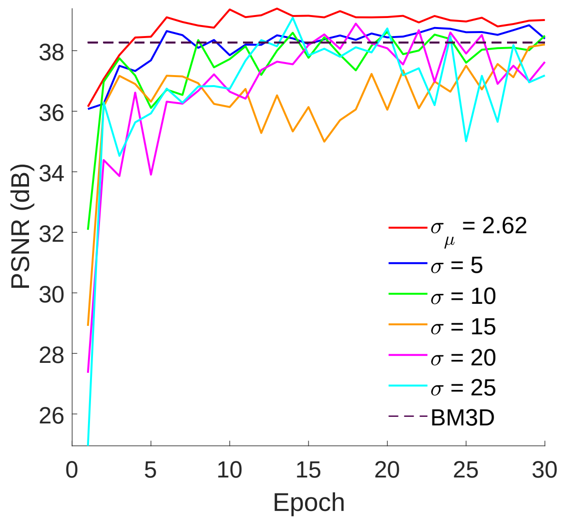
In this chapter, we have addressed a serious need for a high-quality image dataset for denoising research on smartphone cameras. Towards this goal, we have created a public dataset of $\sim 30,000$ images with corresponding high-quality ground truth images for five representative smartphones. We have provided a detailed description on how to capture and process smartphone images to produce this ground truth dataset. Using this dataset, we have benchmarked a number of existing methods to reveal that patch-based methods still outperform learning-based methods trained using conventional ground truthing methods. Our preliminary results on training CNN-based methods using our images (in particular, DnCNN [125]) suggests that CNN-based methods can outperform patch-based methods when trained on proper ground truth images. We believe our dataset and our associated findings will be useful in advancing denoising, especially for methods targeting images captured with smartphones. In the next chapter, we will discuss further research directions in the areas of real image noise modelling.



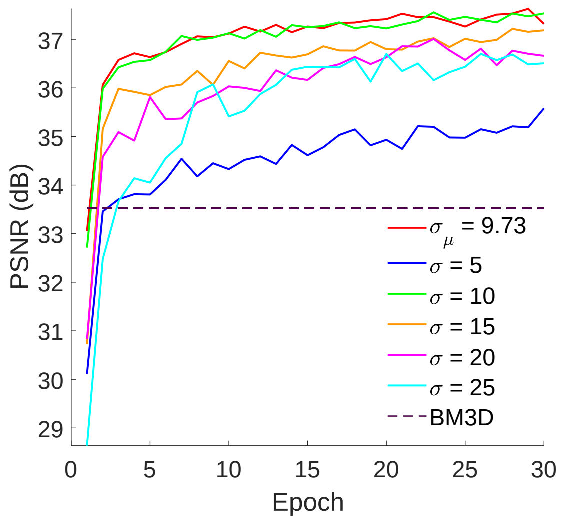
(a) Subset A training



(b) Subset B training



(c) Subset A testing



(d) Subset B testing

Figure 5.12: The effect of using various synthetic noise levels (σ) to train DnCNN [125] on two randomly selected subsets of patches from our dataset, as shown in Table 5.3. The mean noise levels σ_μ on subsets A and B are 2.62 and 9.73, respectively. Using σ values other than σ_μ mostly yields lower performance. One exception is training with $\sigma = 10$ on subset B which yields similar performance as it is so close to σ_μ.

Chapter 6

Noise Flow: Noise Modelling with Conditional Normalizing Flows

Modelling and synthesizing image noise is an important aspect in many computer vision applications; however, as discussed earlier in Chapter 3, the long-standing additive white Gaussian and heteroscedastic (signal-dependent) noise models widely used in the literature provide only a coarse approximation of real sensor noise. In this chapter, we propose our approach for addressing the problem of image noise modelling. We advocate a data-drive approach to develop an accurate and realistic noise model, termed Noise Flow, that is based on recent deep generative models (i.e., normalizing flows [107, 36, 75]). Our approach combines the well-established basic parametric noise models (e.g., signal-dependent noise) with the flexibility and expressiveness of generative normalizing flow networks. First, we will discuss the motivation behind the need for accurate and realistic noise models in Section 6.1. Then, we will discuss some background and related work on noise models and normalizing flows in Section 6.2. Next, we will present our proposed noise model in Section 6.3 followed by experiments and discussion in Section 6.4. Finally, we demonstrate the application of our proposed noise model to the image denoising problem in Section 6.5 and end this chapter in Section 6.6 with conclusions.

6.1 Motivation

Accurately modelling image noise is a critical step towards the final goal of noise reduction in images. As thoroughly discussed in Chapter 3, existing noise models are not sufficient to represent the complexity of real noise [3, 99]. For example, a univariate homoscedastic Gaussian model does not represent the fact that photon noise is signal-dependent—that is, the variance of the noise is proportional to the magnitude of the signal. In turn, the signal-dependent heteroscedastic model [46, 48, 90], often referred to as the noise level function (NLF), does not represent the spatial non-uniformity of noise power (e.g., fixed-pattern noise) or other sources of noise and non-linearities, such as amplification noise and quantization [66]. See Figure 2.1. In spite of their well-known limitations, these models are still the most commonly used. More complex models, such as a Poisson mixture [70, 124], exist, but still do not capture the complex noise sources mentioned earlier.

Contribution In this chapter, we address the aforementioned limitation in existing noise models. We propose a new noise model, termed Noise Flow, that combines the insights of parametric noise models and the expressiveness of powerful generative models. Specifically, we leverage recent normalizing flow architectures [75] to accurately model noise distributions observed from large datasets of real noisy images. In particular, based on the recent Glow architecture [75], we construct a normalizing flow model which is conditioned on critical variables, such as image intensity, camera type, and sensor gain settings (i.e., ISO). The model can be shown to be a strict generalization of the camera NLF but with the ability to capture significantly more complex behaviour. The result is a single model that is compact (fewer then 2500 parameters) and considerably more accurate than existing models. We explore different aspects of the model through a set of ablation studies. To demonstrate the effectiveness of Noise Flow, we consider the application of denoising and use Noise Flow to synthesize training data for a denoising CNN

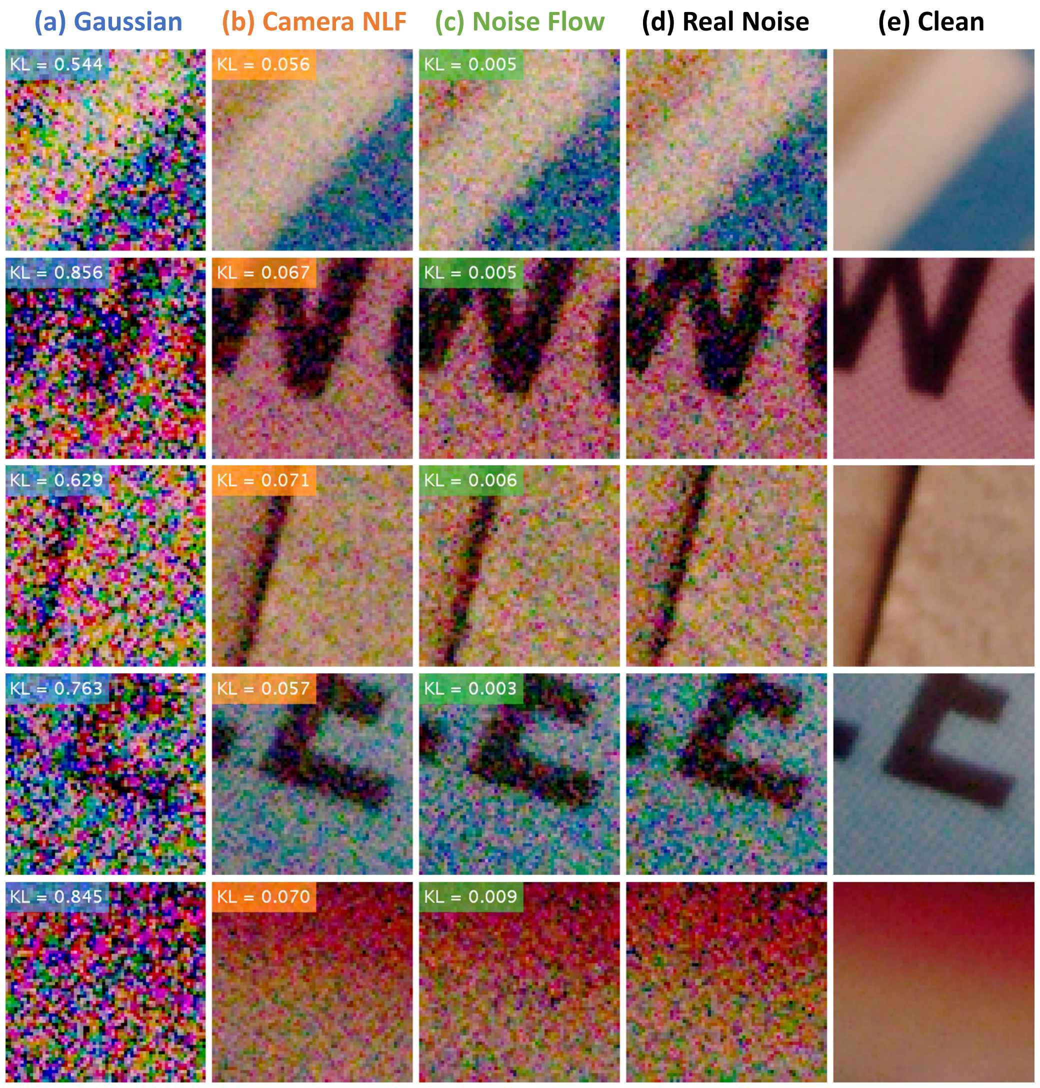


Figure 6.1: Synthetic noisy images generated by (a) a Gaussian model, (b) a heteroscedastic signal-dependent model represented by camera noise level functions (NLF), and (c) our Noise Flow model. Synthetic noise generated from Noise Flow is consistently the most similar to the real noise in (d), qualitatively and quantitatively (in terms of KL divergence relative to the real noise, shown on each image). (e) Reference clean image. Images are from the SIDD [3].

resulting in significant improvements in PSNR.

6.2 Background and Related Work

In this section, we provide a quick recapitulation of existing noise models, which were discussed in more detail in Chapter 3, in order to provide a necessary relationship to our proposed noise model.

Given an observed image $\tilde{\mathbf{I}}$ and its underlying noise-free image $\mathbf{I}$, their relationship can be written as

$$\tilde{\mathbf{I}} = \mathbf{I} + \mathbf{n}, \tag{6.1}$$

where $\mathbf{n}$ is the noise corrupting $\mathbf{I}$. Our focus in this work is to model $\mathbf{n}$.

Several noise models have been proposed in the literature (see Chapter 3). The simplest and most common noise model is the homoscedastic Gaussian assumption, also known as the additive white Gaussian noise (AWGN):

$$n_i \sim \mathcal{N}(0, \sigma^2), \tag{6.2}$$

where n_i is the noise value at pixel i and follows a normal distribution with zero mean and σ^2 variance.

To account for signal dependency of noise, a Poisson distribution $\mathcal{P}$ is used instead as described in Equations 3.4 and 3.5. Neither the Gaussian nor the Poisson models alone can accurately describe image noise. That is because image noise consists of both signal-dependent and signal-independent components. To address such limitation, a Poisson-Gaussian model has been adapted [46, 48, 90], where the noise is a combination of a signal-dependent Poisson distribution and a signal-independent Gaussian distribution as described in Equation 3.6.

A more widely accepted alternative to the Poisson-Gaussian model is to replace the Poisson component by a Gaussian distribution whose variance is signal-dependent [85, 93], which is referred to as the heteroscedastic Gaussian model:

$$n_i \sim \mathcal{N}(0, \alpha^2\, \mathbf{I}_i + \delta^2). \tag{6.3}$$

The heteroscedastic Gaussian model is more commonly referred to as the noise level function (NLF) and describes the relationship between image intensity and noise variance:

$$\mathrm{var}(n_i) = \beta_1\, \mathbf{I}_i + \beta_2, \qquad \beta_1 = \alpha^2, \beta_2 = \delta^2. \tag{6.4}$$

Signal-dependent models may accurately describe noise components, such as photon noise. However, in real images there are still other noise sources that may not be accurately represented by such models [3, 46, 99]. Examples of such sources include fixed-pattern noise, defective pixels, clipped intensities, spatially correlated noise (i.e., cross-talk), amplification, and quantization noise. Some attempts have been made to close the gap between the prior models and the realistic cases of noise—for example, using a clipped heteroscedastic distribution to account for clipped image intensities [46] or using a Poisson mixture model to account for the tail behaviour of real sensor noise [124]. Recently, a generative adversarial network (GAN) was trained for synthesizing noise [26]; however, it was not clear how to quantitatively assess the quality of the generated samples. To this end, there is still a lack of noise models that capture the characteristics of real noise. In the next sections, we propose a data-driven normalizing flow model that can estimate the density of a real noise distribution. Unlike prior attempts, our model can capture the complex characteristics of noise that cannot be explicitly parameterised by existing models.

6.2.1 Normalizing Flows

Normalizing flows were first introduced to machine learning in the context of variational inference [107] and density estimation [36] and are seeing increased interest for generative modelling [75]. A normalizing flow is a transformation of a random variable with a known distribution (typically Normal) through a sequence of differentiable, invertible mappings. Formally, let $\mathbf{x}_0 \in \mathbb{R}^D$ be a random variable with a known and tractable probability density function $p_{\mathcal{X}_0} : \mathbb{R}^D \rightarrow \mathbb{R}$ and let $\mathbf{x}_1, \ldots, \mathbf{x}_N$ be a sequence of random variables such that $\mathbf{x}_i = f_i(\mathbf{x}_{i-1})$ where $f_i : \mathbb{R}^D \rightarrow \mathbb{R}^D$ is a differentiable, bijective function. Then if $\mathbf{n} = f(\mathbf{x}_0) = f_N \circ f_{N-1} \circ \cdots \circ f_1(\mathbf{x}_0)$, the change of variables formula says that the probability density function for $\mathbf{n}$ is

$$p(\mathbf{n}) = p_{\mathcal{X}_0}(g(\mathbf{n})) \prod_{j=1}^{N} \left|\det \mathbf{J}_j(g(\mathbf{n}))\right|^{-1} \ , \tag{6.5}$$

where $g = g_1 \circ \cdots \circ g_{N-1} \circ g_N$ is the inverse of f, and $\mathbf{J}_j = \partial f_j / \partial \mathbf{x}_{j-1}$ is the Jacobian of the jth transformation f_j with respect to its input $\mathbf{x}_{j-1}$ (i.e., the output of f_{j-1}).

Density Estimation A normalizing flow can be directly used for density estimation by finding parameters that maximize the log likelihood of a set of samples. Given the observed data, $\mathcal{D} = \{\mathbf{n}_i\}_{i=1}^{M}$, and assuming the transformations $f_1, \ldots, f_N$ are parameterised by $\Theta = (\theta_1, \ldots, \theta_N)$ respectively, the log likelihood of the data $\log p(\mathcal{D}|\Theta)$ is

$$\sum_{i=1}^{M} \log p_{\mathcal{X}_0}(g(\mathbf{n}_i|\Theta)) - \sum_{j=1}^{N} \log \left|\det \mathbf{J}_j(g(\mathbf{n}_i|\Theta), \theta_j)\right| \ , \tag{6.6}$$

where the first term is the log likelihood of the sample under the base measure and the second term, sometimes called the log-determinant or volume correction, accounts for the change of volume induced by the transformation by the normalizing flows.

Bijective Transformations To construct an efficient normalizing flow we need to define differentiable and bijective transformations f. Beyond being able to define and compute f, we also need to be able to efficiently compute its inverse, g, and the log determinant $\log|\det \mathbf{J}|$, which are necessary to evaluate the data log likelihood in Equation 6.6. First, consider the case of a linear transformations [75]

$$f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b} \ , \tag{6.7}$$

where $\mathbf{A} \in \mathbb{R}^{D \times D}$ and $\mathbf{b} \in \mathbb{R}^{D}$ are parameters. For f to be invertible $\mathbf{A}$ must have full rank; its inverse is given by $g(\mathbf{x}) = \mathbf{A}^{-1}(\mathbf{x} - \mathbf{b})$ and the determinant of the Jacobian is simply $\det \mathbf{J} = \det \mathbf{A}$.

Affine Coupling To enable more expressive transformations, we can use the concept of coupling [36]. Let $\mathbf{x} = (\mathbf{x}_A, \mathbf{x}_B)$ be a disjoint partition of the dimensions of $\mathbf{x}$ and let $\hat{f}(\mathbf{x}_A|\theta)$ be a bijection on $\mathbf{x}_A$ that is parameterised by θ. Then a coupling flow is

$$f(\mathbf{x}) = (\hat{f}(\mathbf{x}_A; \theta(\mathbf{x}_B)), \mathbf{x}_B) \ , \tag{6.8}$$

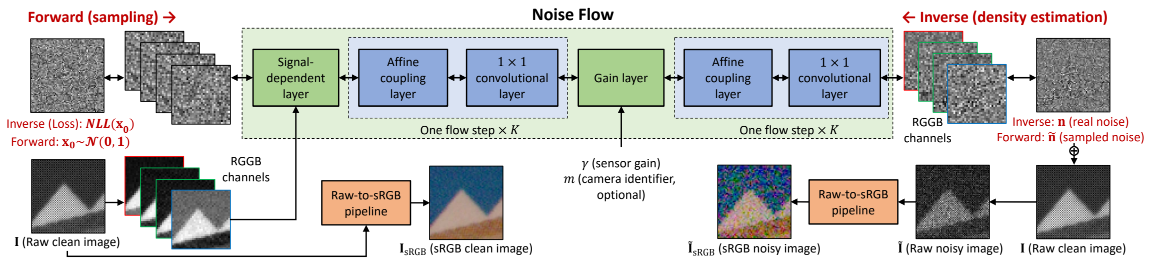
where $\theta(\mathbf{x}_B)$ is *any* arbitrary function which uses only $\mathbf{x}_B$ as input. The power of a coupling flow resides, largely, in the ability of $\theta(\mathbf{x}_B)$ to be arbitrarily complex. For instance, shallow ResNets [63] were used for this function in [75].

Inverting a coupling flow can be done by using the inverse of $\hat{f}$. Further, the Jacobian of f is a block triangular matrix where the diagonal blocks are $\hat{\mathbf{J}}$ and the identity. Hence, the determinant of the Jacobian is simply the determinant of $\hat{\mathbf{J}}$. A common form of a coupling layer is the *affine coupling layer* [37, 75]

$$\hat{f}(\mathbf{x}; \mathbf{a}, \mathbf{b}) = \mathbf{D}\mathbf{x} + \mathbf{b} \tag{6.9}$$

where $\mathbf{D} = \text{diag}(\mathbf{a})$ is a diagonal matrix. To ensure that $\mathbf{D}$ is invertible and has non-zero diagonals it is common to use $\mathbf{D} = \text{diag}(\exp(\mathbf{a}))$.

With the above formulation of normalizing flows, it becomes clear that we can utilize their expressive power for modelling real image noise distributions and mapping them to easily tractable simpler distributions. As a by-product, such models can directly be used for realistic noise synthesis. Since the introduction of normalizing flows to machine learning, they have been focused towards image generation tasks (e.g., [75]). However, in this work, we adapt normalizing flows to the task of noise modelling and synthesis by introducing two new conditional bijections, that we describe next.



Forward (sampling) →
Noise Flow
← Inverse (density estimation)
Signal-dependent layer
Affine coupling layer
1 × 1 convolutional layer
Gain layer
Affine coupling layer
1 × 1 convolutional layer
One flow step × K
One flow step × K
Inverse (Loss): $NLL(\mathbf{x}_0)$
Forward: $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$
RGGB channels
RGGB channels
Inverse: $\mathbf{n}$ (real noise)
Forward: $\tilde{\mathbf{n}}$ (sampled noise)
γ (sensor gain)
m (camera identifier, optional)
Raw-to-sRGB pipeline
Raw-to-sRGB pipeline
$\mathbf{I}$ (Raw clean image)
$\mathbf{I}_{\text{sRGB}}$ (sRGB clean image)
$\tilde{\mathbf{I}}_{\text{sRGB}}$ (sRGB noisy image)
$\tilde{\mathbf{I}}$ (Raw noisy image)
$\mathbf{I}$ (Raw clean image)

Figure 6.2: The architecture of our Noise Flow model. The affine coupling and 1×1 convolutional layers are ported from [75]. The signal-dependent and gain layers are newly proposed. The Raw-to-sRGB pipeline is ported from [3].

6.3 Noise Flow

In this section, we propose our new architecture of normalizing flows for modelling noise that we call Noise Flow. Noise Flow contains novel bijective transformations that capture the well-established and fundamental aspects of parametric noise models (e.g., signal-dependent noise and gain) which are mixed with more expressive and general affine coupling transformations.

6.3.1 Noise Modelling using Normalizing Flows

Starting from Equations 6.1 and 6.6, we can directly use normalizing flows to estimate the probability density of a complex noise distribution. Let $\mathcal{D} = \{\mathbf{n}_i\}_{i=1}^{M}$ denote a dataset of observed camera noise where $\mathbf{n}_i$ is the noise layer corrupting a raw-RGB image. Noise layers can be obtained by subtracting a clean image from its corresponding noisy one. As is common, we choose an isotropic Normal distribution with zero mean and identity covariance as the base measure. Next, we choose a set of bijective transformations, with a set of parameters Θ, that define the normalizing flows model. Lastly, we train the model by minimizing the negative log likelihood of the transformed distribution, as indicated in Equation 6.6.

We choose the Glow model [75] as our starting point. We use two types of bijective transformations (i.e., layers) from the Glow model: (1) the affine coupling layer as defined in Equation 6.9 that can capture arbitrary correlations between image dimensions (i.e., pixels); and (2) the 1×1 convolutional layers that are used to capture cross-channel correlations in the input images.

6.3.2 Noise Modelling using Conditional Normalizing Flows

Existing normalizing flows are generally trained in an unsupervised manner using only data samples and without additional information about the data. In our case, we have some knowledge regarding the noise types, such as the signal-dependency of noise and the scaling of the noise based on sensor gain. Some of these noise types are shown in Figure 2.1 along with their associated imaging processes. Thus, we propose new normalizing flow layers that are conditional on such information. However, many noise types, such as fixed-pattern noise, cannot be easily specified directly. To capture these other phenomena we use a combination of affine coupling layers (Equations 6.8 and 6.9) and 1×1 convolutional layers (a form of Equation 6.7) which were introduced by the Glow model [75].

Figure 6.2 shows the proposed architecture of our noise model (Noise Flow). Noise Flow is a sequence of a signal-dependent layer; K unconditional flow steps; a gain layer; and another set of K unconditional flow steps. Each unconditional flow step is a block of an affine coupling layer followed by a 1×1 convolutional layer. The term K is the number of flow steps to be used in the model. In our experiments, we use $K = 4$, unless otherwise specified. We tried several values for K ranging between 2 and 64, and found that $K = 4$ gives a good balance between the negative log likelihood (NLL) and the number of parameters in the model. The model is fully bijective—that is, it can operate in both directions, meaning that it can be used for both simulating noise (by sampling from the base measure $\mathbf{x}_0$ and applying the sequence of transformations) or likelihood evaluation (by using the inverse transformation given a noise sample $\tilde{\mathbf{I}}$ to evaluation of Equation 6.5). The Raw-to-sRGB rendering pipeline is imported from [3]. Next, we discuss the proposed signal-dependent and gain layers in detail.

Signal-Dependent Layer

We construct a bijective transformation that mimics the signal-dependent noise defined in Equation 6.3. This layer is defined as

$$f(\mathbf{x}) = \mathbf{s} \odot \mathbf{x}, \qquad \mathbf{s} = (\beta_1 \mathbf{I} + \beta_2)^{\frac{1}{2}}. \tag{6.10}$$

The inverse of this layer is given by $g(\mathbf{x}) = \mathbf{s}^{-1} \odot \mathbf{x}$, where $\mathbf{I}$ is the latent clean image, and $\odot$ is point-wise multiplication. To account for volume change induced by this transformation, we compute the log determinant as

$$\log |\det \mathbf{J}| = \sum_{i=1}^{D} \log(s_i) \tag{6.11}$$

where s_i is the ith element of $\mathbf{s}$ and D is the dimensionality (i.e., number of pixels and channels) of $\mathbf{x}$. The signal-dependent noise parameters β_1 and β_2 should be strictly positive as the standard deviation of noise should be positive and an increasing function of intensity. Thus, we parameterise them as $\beta_1 = \exp(b_1)$ and $\beta_2 = \exp(b_2)$. We initialize the signal-dependent layer to resemble an identity transformation by setting $b_1 = -5.0$ and $b_2 = 0$. This way, $\beta_1 \approx 0$ and $\beta_2 = 1.0$, and hence the initial scale $\mathbf{s} \approx 1.0$. Having $\mathbf{s} \approx 1.0$ makes the signal-dependent layer work as an identity transformation, which is a convenient starting point for training that layer.

Gain Layer

Sensor gain amplifies not only the signal, but also the noise. With common use of higher gain factors in low-light imaging, it becomes essential to explicitly factor the effect of gain in any noise model. Hence, we propose a gain-dependent bijective transformation as a layer of Noise Flow. The gain layer is modelled as a scale factor γ of the corresponding

ISO level of the image, and hence the transformation is

$$f(\mathbf{x}) = \gamma(\text{ISO}) \odot \mathbf{x}, \qquad \gamma(\text{ISO}) = u(\text{ISO}) \times \text{ISO}, \tag{6.12}$$

where $u(\text{ISO}) > 0$ allows the gain factors to vary somewhat from the strict scaling dictated by the ISO value. The inverse transformation is $g(\mathbf{x}) = \gamma^{-1}(\text{ISO}) \odot \mathbf{x}$, where u is parameterised to be strictly positive and is initialized to $u \approx 1/200$ to account for the typical scale of the ISO values. Finally, the log determinant of this layer is

$$\log |\det \mathbf{J}| = D \log(\gamma(\text{ISO})), \tag{6.13}$$

where D is the number of dimensions (i.e., pixels and channels) in $\mathbf{x}$. There are many ways to represent $u(\text{ISO})$. However, since the available dataset contained only a small set of discrete ISO levels, we chose to simply use a discrete set of values. Formally $u(\text{ISO}) = \exp(v_{\text{ISO}})$ where the exponential is used to ensure that $u(\text{ISO})$ is positive. We use a single parameter for each ISO level in the dataset (e.g., $\{v_{100}, \ldots, v_{1600}\}$). The values of v_{ISO} are initialized so that $\exp(v_{\text{ISO}}) \approx 1/200$ to account for the scale of the ISO value and ensure the initial transformation remains close to an identity transformation.

Different cameras may have different gain factors corresponding to their ISO levels. These camera-specific gain factors are usually proprietary and hard to access but may have a significant impact on the noise distribution of an image. To handle this, we use an additional set of parameters to adjust the gain layer for each camera. In this case, the above gain layer is adjusted by introducing a camera-specific scaling factor. That is,

$$\gamma(\text{ISO}, m) = \psi_m \times u(\text{ISO}) \times \text{ISO}, \tag{6.14}$$

where $\psi_m \in \mathbb{R}^+$ is the scaling factor for camera m. This is a simple model but was found to be effective to capture differences in gain factors between cameras.

6.4 Experiments

To assess the performance of Noise Flow, we train it to model the realistic noise distribution of our Smartphone Image Denoising Dataset (SIDD) from Chapter 5 and also evaluate the sampling accuracy of the trained model.

6.4.1 Experimental Setup

Dataset We chose SIDD for training Noise Flow as it best fits our task for noise modelling, mainly due to the great extent of variety in cameras, ISO levels, and lighting conditions. A more detailed discussion of the SIDD is provided in Chapter 5.

Data preparation We start by collecting a large number of realistic noise samples from the SIDD. We obtain the noise layers by subtracting the ground truth images from the noisy ones. In our experiments, we use only raw-RGB images as they directly represent the noise distribution of the underlying cameras. We avoid using sRGB images as rendering image into sRGB space tends to significantly change the noise distribution [94]. We arrange the data as approximately $500,000$ image patches of size 64×64 pixels. We split the data into a training set $\mathcal{D}_\mathrm{r}$ of approximately 70% of the data and a testing set $\mathcal{D}_\mathrm{s}$ of approximately 30% of the data. We ensure that the same set of cameras and ISO levels is represented in both the training and testing sets. For visualization only, we render raw-RGB images through a colour processing pipeline into sRGB colour space.

The SIDD provides only the gain amplified clean image $\mathbf{I}_\gamma$ and not the latent clean image $\mathbf{I}$. We use the learned gain parameter γ to correct for this mismatch and estimate the latent clean image as $\mathbf{I} = \mathbf{I}_\gamma / \gamma$ when it is needed in the signal-dependant layer.

Loss function and evaluation metrics We train Noise Flow as a density estimator of the noise distribution of the dataset which can be also used to generate noise samples from this distribution. For density estimation training, we use the negative log likelihood (NLL) of the training set (see Equation 6.6) as the loss function which is optimized using

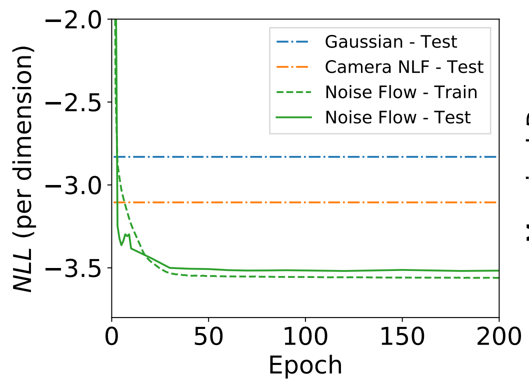
Adam [74] with learning rate 10^{-3} and exponential decay rate for the first and second moment estimates 0.9, and 0.999, respectively. For evaluation, we consider the same NLL evaluated on the test set.

To provide further insight in the differences between the approaches, we also consider the Kullback-Leibler (KL) divergence of the pixel-wise marginal distributions between generated samples and test set samples. Such a measure ignores the ability of a model to capture correlations but focuses on a model's ability to capture the most basic characteristics of the distribution. Specifically, given an image from the test set, we generate a noise sample from the model and compute histograms of the noise values from the test image and the generated noise and report the discrete KL divergence between the histograms.

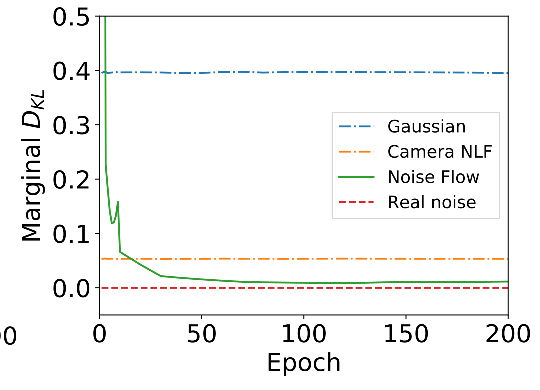
Baselines We compare the Noise Flow models against two well-established baseline models. The first is the homoscedastic Gaussian noise model (i.e., AWGN) defined in Equation 6.2. We prepare this baseline model by estimating the maximum likelihood estimate (MLE) of the noise variance of the training set, assuming a univariate Gaussian distribution. The second baseline model is the heteroscedastic Gaussian noise model (i.e., NLF), described in Equations 6.3 and 6.4, as provided by the camera devices. The SIDD provides the camera-calibrated NLF for each image. We use these NLFs as the parameters of the heteroscedastic Gaussian model for each image. During testing, we compute the NLL of the testing set against both baseline models.

6.4.2 Results and Ablation Studies

Noise Density Estimation Figure 6.3a shows the training and testing NLL on the SIDD of Noise Flow compared to (1) the Gaussian noise model and (2) the signal-dependent noise model as represented by the camera-estimated noise level functions (NLFs). It is clear that Noise Flow can model the realistic noise distribution better than Gaussian and signal-dependent models. As shown in Table 6.1, Noise Flow achieves the best NLL, with



(a)



(b)

Figure 6.3: (a) NLL per dimension on the training and testing sets of Noise Flow compared to (1) the Gaussian model and (2) the signal-dependent model as represented by the camera-estimated NLFs. (b) Marginal KL divergence (D_{KL}) between the generated and the real noise samples.

	Gaussian	Cam. NLF	Noise Flow
NLL	-2.831 (99.4%)	-3.105 (51.6%)	$\mathbf{-3.521}$
D_{KL}	0.394 (97.9%)	0.052 (84.1%)	$\mathbf{0.008}$

Table 6.1: Best achieved testing NLL and marginal D_{KL} for Noise Flow compared to the Gaussian and Camera NLF baselines. Relative improvements of Noise Flow on other baselines, in terms of *likelihood*, are in parentheses.

0.69 and 0.42 nats/pixel improvement over the Gaussian and camera NLF models, respectively. This translates to 99.4% and 51.6% improvement in likelihood, respectively. We calculate the improvement in likelihood by calculating the corresponding improvement in $\exp(-NLL)$.

Noise Synthesis Figure 6.3b shows the average marginal KL divergence between the generated noise samples and the corresponding noise samples from the testing set for the three models: Gaussian, camera NLF, and Noise Flow. Noise Flow achieves the best KL divergence, with 97.9% and 84.1% improvement over the Gaussian and camera NLF models, respectively, as shown in Table 6.1.

Figure 6.4 shows generated noise samples from Noise Flow compared to samples from

Gaussian and camera NLF models. We show samples from various ISO levels in the range $\{100, \ldots, 1600\}$ and lighting conditions (N: normal light, L: low light). Noise Flow samples are the closest to the real noise distribution in terms of the marginal KL divergence. Also, there are more noticeable visual similarities between Noise Flow samples and the real samples compared to the Gaussian and camera NLF models.

Learning signal-dependent noise parameters Figure 6.5a shows the learning of the signal-dependent noise parameters β_1 and β_2 as defined in Equation 6.4 while training a Noise Flow model. The parameters are converging towards values that are consistent with the signal-dependent noise model where β_1 is the dominant factor that represents the Poisson component of the noise and β_2 is the smaller factor representing the additive Gaussian component of the noise. In our experiments, these parameters are run through an exponential function to force their values to be strictly positive.

Learning gain factors Figure 6.5b shows the learning of the gain factors as defined in Equation 6.12 while training a Noise Flow model. The gain factors $\{\gamma_{100}, \ldots, \gamma_{1600}\}$ are consistent with the corresponding ISO levels indicated by their subscripts. This shows the ability of the Noise Flow model to properly factor the sensor gain in the noise modelling and synthesis process. Note that we omitted ISO level 200 from the training and testing sets because there are not enough images from this ISO level in the SIDD.

Learning camera-specific parameters In our Noise Flow model, the camera-specific parameters consist of a set of gain scale factors $\{\psi_m\}$, one for each of the five cameras in the SIDD. Figure 6.6 shows these gain scales for each camera in the dataset during the course of training. It is clear that there are differences between cameras in the learned gain behaviours. These differences are consistent with the differences in the noise level function parameter β_1 of the corresponding cameras shown in Figure 6.6b and capture fundamental differences in the noise behaviour between devices. This demonstrates the importance of the camera-specific parameters to capture camera-specific noise profiles. Training Noise Flow for a new camera can be done by fine-tuning the camera-specific

Model	NLL	D_{KL}
S-G	-3.431 (9.42%)	0.067 (88.1%)
S-G-CAM	-3.511 (1.01%)	0.010 (20.0%)
S-Ax1-G-Ax1-CAM	-3.518 (0.30%)	0.009 (11.1%)
S-Ax4-G-Ax4-CAM (Noise Flow)	$\mathbf{-3.521}$	**0.008**

Table 6.2: Best achieved testing NLL and marginal D_{KL} for different layer architectures. The symbols S, G, CAM, Ax1, and Ax4 indicate a signal layer, gain layer, camera-specific parameters, one unconditional flow step, and four unconditional flow steps, respectively. Relative improvements of Noise Flow, in terms of likelihood, are in parentheses.

parameters within the gain layers; all other layers (i.e., the signal-dependent and affine coupling layers) can be considered non-camera-specific.

Effect of individual layers Table 6.2 compares different architecture choices for our Noise Flow model. We denote the different layers as follows: G: gain layer; S: signal-dependent layer; CAM: a layer using camera-specific parameters; Ax1: one unconditional flow step (an affine coupling layer and a 1×1 convolutional layer); Ax4: four unconditional flow steps. The results show a significant improvement in noise modelling (in terms of NLL and D_{KL}) resulting from the additional camera-specific parameters (i.e., the S-G-CAM model), confirming the differences in noise distributions between cameras and the need for camera-specific noise parameters. Then, we show the effect of using affine coupling layers and 1×1 convolutional layers in our Noise Flow model. Adding the Ax1 blocks improves the modelling performance in terms of NLL. Also, increasing the number of unconditional flow steps from one to four introduces a slight improvement as well. This indicates the importance of affine coupling layers in capturing additional pixel-correlations that cannot be directly modeled by the signal-dependency or the gain layers. The S-Ax4-G-Ax4-CAM is the final Noise Flow model.

6.5 Application to Real Image Denoising

Preparation To further investigate the accuracy of the Noise Flow model, we use it as a noise generator to train an image denoiser. We use the DnCNN image denoiser [125]. We use the clean images from the SIDD-Medium [3] as training ground truth and the SIDD-Validation as our testing set. The SIDD-Validation contains both real noisy images and the corresponding ground truth. We compare three different cases for training DnCNN using synthetically generated noise: (1) DnCNN-Gauss: homoscedastic Gaussian noise (i.e., AWGN); (2) DnCNN-CamNLF: signal-dependent noise from the camera-calibrated NLFs; and (3) DnCNN-NF: noise generated from our Noise Flow model. For the Gaussian noise, we randomly sample standard deviations from the range $\sigma \in [0.24, 11.51]$. For the signal-dependent noise, we randomly select from a set of camera NLFs. For the noise generated with Noise Flow, we feed the model with random camera identifiers and ISO levels. The σ range, camera NLFs, ISO levels, and camera identifiers are all reported in the SIDD. Furthermore, in addition to training with synthetic noise, we also train the DnCNN model with real noisy/clean image pairs from the SIDD-Medium and no noise augmentation (indicated as DnCNN-Real).

Results and discussion Table 6.3 shows the best achieved testing peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) [118] of DnCNN using the aforementioned three noise synthesis strategies and the discriminative model trained on real noise. The model trained on noise generated from Noise Flow yields the highest PSNR and SSIM values, even slightly higher than DnCNN-Real due to the relatively limited number of samples in the training dataset. We also report, in parentheses, the relative improvement introduced by DnCNN-NF over the other two models in terms of root-mean-square-error (RMSE) and structural dissimilarity (DSIMM) [87, 120], for PSNR and SSIM, respectively. We preferred to report relative improvement in this way because PSNR and SSIM tend to saturate as errors get smaller; conversely, RMSE and DSSIM do not saturate. For visual inspection, in Figure 6.7, we show some denoised images from the best trained model from the three cases, along with the corresponding noisy and clean images. DnCNN-Gauss tends to over-smooth noise, as in rows 3 and 5, while DnCNN-CamNLF frequently causes artifacts and pixel saturation, as in rows 1 and 5. Although DnCNN-NF does not consistently yield the highest PSNR, it is the most stable across all six images.

Figure 6.8 shows the training loss over 2000 epochs. Despite that training loss of the DnCNN-CamNLF is the lowest, the training behaviour indicated that DnCNN-NoiseFlow is the most stable. The DnCNN-Gauss model is also stable, however, it still yields lower testing performance, as shown in Figure 6.9.

Figure 6.9 shows the testing peak signal-to-noise ratio (PSNR) over 2000 epochs of the three models from Figure 6.8. DnCNN-CamNLF outperforms the DnCNN-Gauss models by a small margin, despite the lower training loss of the former. The DnCNN-NoiseFlow model yields the best performance despite having slightly higher training loss than DnCNN-CamNLF.

Figure 6.10 shows more denoising results from the three models for visual inspection. The samples confirm the better performance of the DnCNN-NoiseFlow model and the importance of having more accurate noise models for generating realistic synthetic noise.

Noise Flow [2] can be used beyond image denoising in assisting computer vision tasks that require noise synthesis (e.g., robust image classification [35] and burst image deblurring [8]. In addition, Noise Flow would give us virtually unlimited noise samples compared to the limited numbers in the datasets.

6.6 Conclusion

In this chapter, we have presented a conditional normalizing flow model for image noise modelling and synthesis that combines well-established noise models and the expressiveness of normalizing flows. As an outcome, we provide a compact noise model with fewer

Model	PSNR	SSIM
DnCNN-Gauss	43.63 (43.0%)	0.968 (75.6%)
DnCNN-CamNLF	44.99 (33.4%)	0.982 (56.0%)
DnCNN-NF	**48.52**	**0.992**
DnCNN-Real	47.08 (15.3%)	0.989 (27.5%)

Table 6.3: DnCNN denoiser [125] trained on synthetic noise generated with Noise Flow (DnCNN-NF) achieves higher PSNR and SSIM values compared to training on synthetic noise, from a Gaussian model or camera NLFs, and real noise. Relative improvements of DnCNN-NF over other models, in terms of RMSE and DSSIM, are in parentheses.

than 2500 parameters that can accurately model and generate realistic noise distributions with 0.42 nats/pixel improvement (i.e., 52% higher likelihood) over camera-calibrated noise level functions. We believe the proposed method and the provided model will be very useful for advancing many computer vision and image processing tasks.

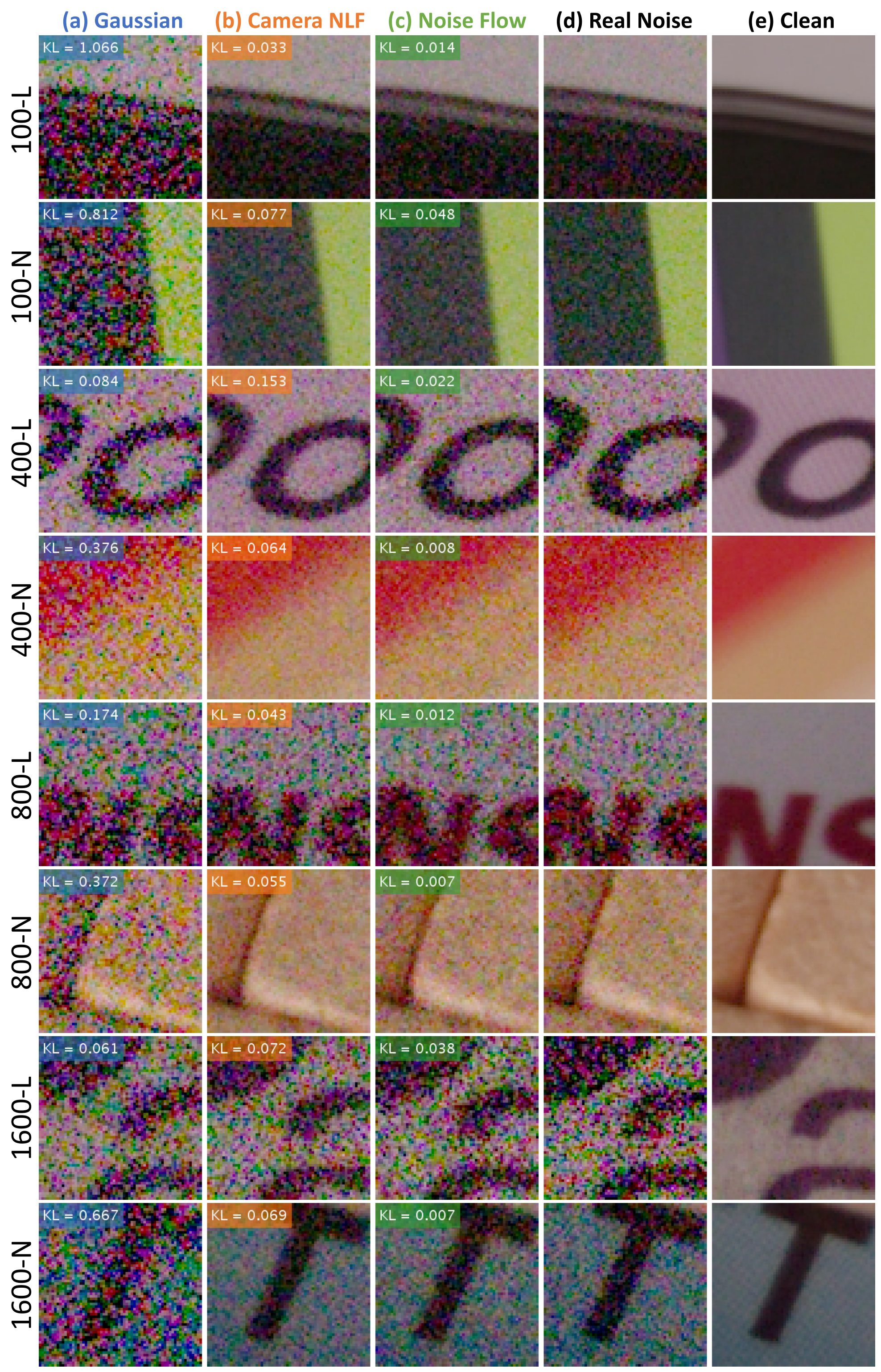
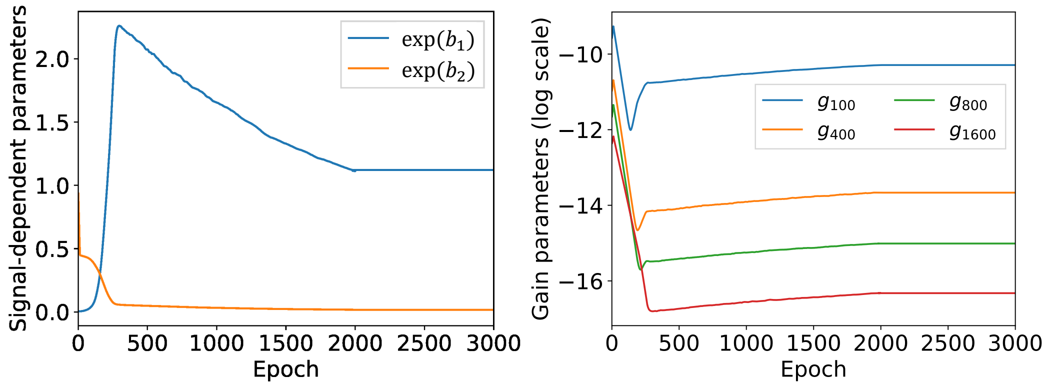
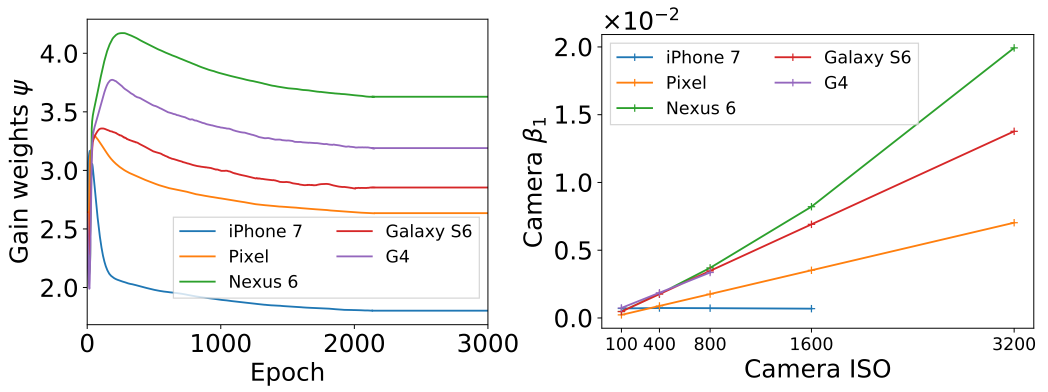


Figure 6.4: Generated noise samples from (c) Noise Flow are much closer, in terms of marginal KL divergence, to (d) the real samples; compared to (a) Gaussian and (b) camera NLF models. (e) Clean image. Corresponding ISO levels and lighting conditions are on the left.



(a) Signal-dependent parameters (b) Gain parameters

Figure 6.5: (a) Signal-dependent noise parameters β_1 and β_2 are consistent with the signal-dependent noise model where β_1 is dominant and β_2 is much smaller. (b) The gain parameters, in log scale, are consistent with the corresponding ISO levels shown in the legend.



(a) Camera gain weights (b) Camera β_1 values

Figure 6.6: (a) Learned camera-specific weights for the shared gain layer indicates differences in the gain of different cameras. These gains are correlated with the cameras' different values for NLF parameter β_1 shown in (b). The iPhone and G4 cameras have smaller ranges of ISO values and hence their correlation with the gains is not clear.

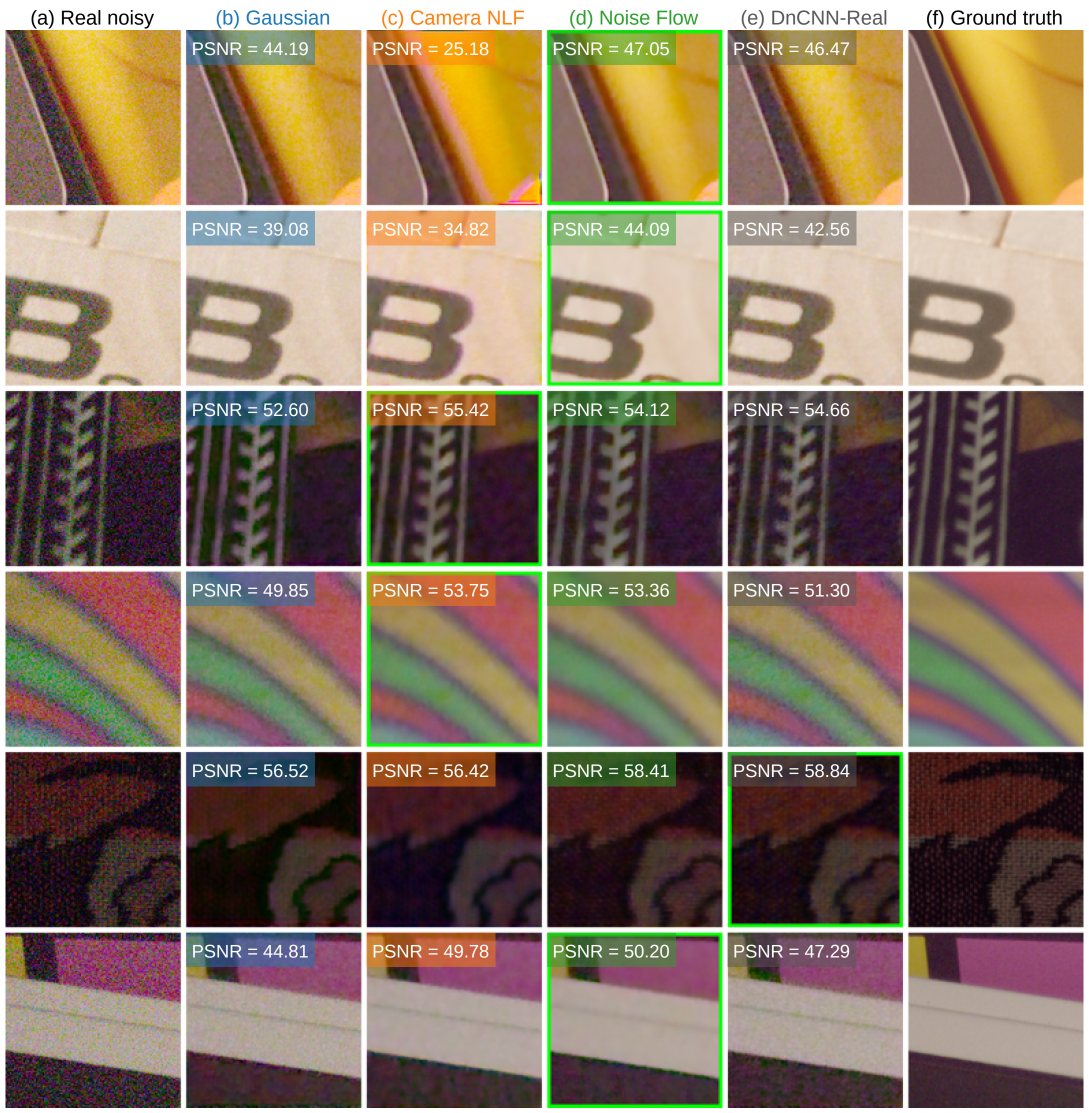


Figure 6.7: Sample denoising results from DnCNN trained on three different noise synthesis methods: (b) Gaussian; (c) camera NLF; and (d) Noise Flow. (e) DnCNN trained on real noise. (a) Real noisy image. (f) Ground truth.

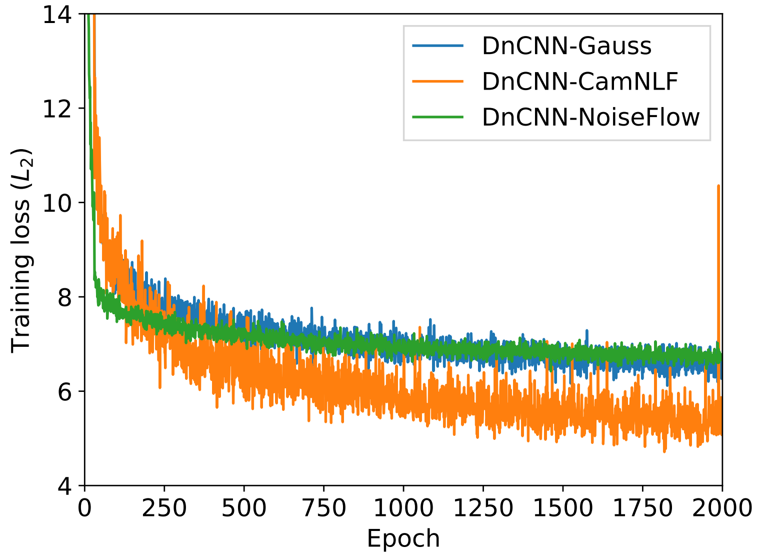


Figure 6.8: Results of training the DnCNN [125] image denoiser with synthetic noise generated by: the Gaussian model; the camera NLFs; and our Noise Flow model.

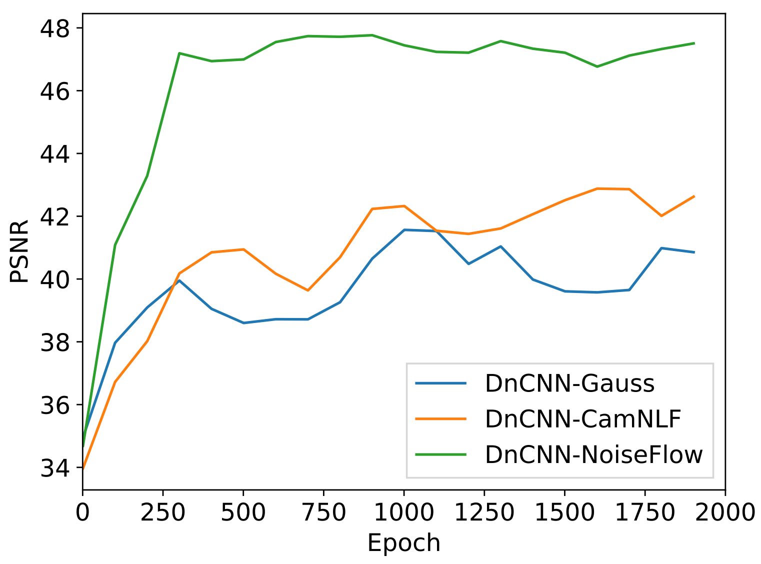


Figure 6.9: Testing PSNR results corresponding to the three model from Figure 6.8. The DnCNN model trained on noise generated with Noise Flow yields the best PSNRs.

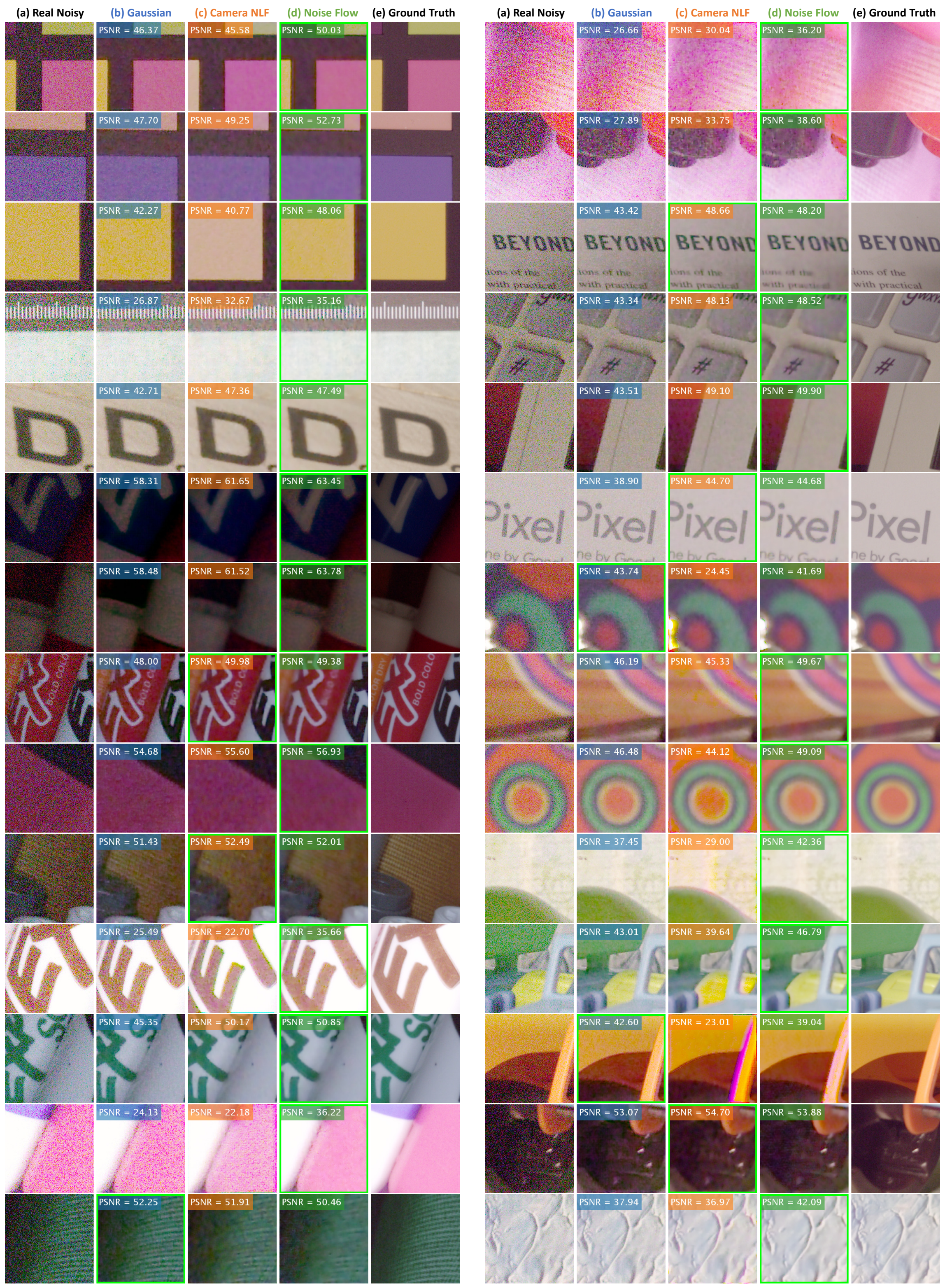


Figure 6.10: Sample denoising results from the DnCNN denoiser trained on three different noise synthesis methods:(b) Gaussian; (c) camera NLF; and (d) Noise Flow. (a) Real noisy image. (e) Ground truth.

Chapter 7

Benchmarking Denoising Algorithms on Real Noisy Images

In this chapter, we present the NTIRE real image denoising challenge that is based on our SIDD benchmark previously discussed in Chapter 5. This challenge has been hosted at the New Trends in Image Restoration and Enhancement (NTIRE) Workshop in the years 2019 and 2020, and it has attracted many researchers in the areas of image denoising and noise modelling. This challenge has two tracks for quantitatively evaluating image denoising performance in (1) the Bayer-pattern raw-RGB and (2) the standard RGB (sRGB) colour spaces.

7.1 Motivation

Image denoising is a fundamental and active research area (e.g., [113, 125, 126, 56]) with a long-standing history in computer vision (e.g., [77, 82]). A primary goal of image denoising is to remove or correct for noise in an image, either for aesthetic purposes, or to help improve other downstream tasks. For many years, researchers have primarily relied on synthetic noisy images for developing and evaluating image denoisers, especially additive white Gaussian noise (AWGN)—for example, [20, 32, 125]. Recently, more focus has been given to evaluating image denoisers on real noisy images [3, 99]. It was shown that the performance of learning-based image denoisers on real noisy images can be limited if trained using only synthetic noise. Also, hand-engineered and statistics-based methods have been shown to perform better on real noisy images. To this end, we have proposed this challenge as a means to evaluate and benchmark image denoisers on real noisy images.

Contribution This challenge is based on our Smartphone Image Denoising Dataset (SIDD) [3] that consists of thousands of real noisy images with their estimated ground-truth, in both raw sensor data (raw-RGB) and standard RGB (sRGB) colour spaces. Hence, in this challenge, we provide two challenge tracks for benchmarking image denoisers in both raw-RGB and sRGB colour spaces. This challenge is aimed to serve as a unified and publicly available framework for researchers to evaluate their image denoising methods. As part of the efforts done in preparing such a framework, we provide tools to facilitate the development of image denoisers in the raw-RGB color space, such as a software-based camera rendering pipeline and camera metadata extraction tools. We present more details on the challenge framework in the next section.

7.2 The Challenge

The NTIRE Real Image Denoising Challenge is aimed to gauge and advance the state-of-the-art in image denoising. The focus of the challenge is on evaluating image denoisers on *real*, rather than synthetic, noisy images. In the following, we present some details about the dataset used in the challenge and how the challenge is designed.

7.2.1 Dataset

We used the SIDD dataset [3] introduced in Chapter 5 to provide training, validation, and testing images for the challenge. As previously mentioned, the SIDD dataset consists of thousands of real noisy images and their corresponding ground truth, from ten different scenes, captured repeatedly with five different smartphone cameras under different lighting conditions and ISO levels. The ISO levels ranged from 50 to 10,000. The images are provided in both raw-RGB and sRGB colour spaces. We believe this dataset is a good fit for benchmarking image denoisers on real noisy images, mainly due to the great extent of variety in noise levels and lighting conditions found in the dataset. Also, this dataset is large enough to provide sufficient training data for learning-based methods, especially, convolutional neural networks (CNNs). More details about the SIDD are presented in Chapter 5.

7.2.2 Challenge Design and Tracks

Tracks We provide two tracks to benchmark the proposed image denoisers based on two different colour spaces: the raw-RGB and the sRGB. Images in the raw-RGB format represent minimally processed images obtained directly from the camera's sensor. These images are in a sensor-dependent colour space where the R, G, and B values are related to the sensor's colour filter array's spectral sensitivity to incoming visible light. Images in the sRGB format represent the camera's raw-RGB image that have been processed by the in-camera image processing pipeline to map the sensor-dependent RGB colours to a device-independent colour space, namely standard RGB (i.e., sRGB). Different camera models apply their own proprietary photo-finishing routines, including several nonlinear colour manipulations, to modify the raw-RGB values to appear visually appealing (see [71] for more details). We note that the provided sRGB images are not compressed and therefore do not exhibit compression artifacts. Denoising a raw-RGB would typically represent a denoising module applied within the in-camera image processing pipeline. Denoising an sRGB image would represent a denoising module applied after the in-camera colour manipulation. As shown in Chapter 5 image denoisers tend to perform better in the raw-RGB colour space than in the sRGB colour space. However, raw-RGB images are far less common than sRGB images, which are easily saved in common formats, such as JPEG and PNG. Since the SIDD dataset contains both raw-RGB and sRGB versions of the same image, we found it feasible to provide a separate track for denoising in each colour space. Both tracks follow similar data preparation, evaluation, and competition timeline, as discussed next.

Training data The provided training data was the SIDD-Medium dataset that consists of 320 noisy images in both raw-RGB and sRGB space with corresponding ground truth and metadata. Each noisy or ground truth image is a 2D array of normalized raw-RGB values (mosaiced colour filter array) in the range $[0, 1]$ in single-precision floating point format saved as Matlab .mat files. The metadata files contained dictionaries of Tiff tags for the raw-RGB images, saved as .mat files.

Validation and testing data We provide two different validation and testing datasets for the 2019 and 2020 versions of the challenge. For the 2019 version, the validation and testing data were the same as the SIDD benchmark data. The SIDD benchmark validation data consisted of 1280 noisy image blocks (i.e., croppings) form both raw-RGB and sRGB images, each block is 256×256 pixels. The blocks are taken from 40 images, 32 blocks from each image ($40 \times 32 = 1280$). All image blocks are combined in a single 4D array of shape $[40, 32, 256, 256]$ where the four dimensions represent the image index, the index of the block within the image, the block height, and the block width, respectively. The blocks have the same number format as the training data. The testing data consisted of 1280 noisy image blocks different from the validation block but following the same format as the validation data. Image metadata files were also provided for the 40 images from which the validation/testing data was extracted.

For the 2020 version of the challenge, we captured an entirely new set of images with a new set of smartphone cameras: Google Pixel 2, Google Pixel 3, LG G7+, Apple iPhone7, Apple iPhone X, and HTC U12+. We used this new image set to generate new validation and testing datasets following a similar procedure to the one used in the SIDD benchmark, and hence, we named this new dataset SIDD+. The SIDD+ validation set consists of 1024 noisy image blocks (i.e., croppings) form both rawRGB and sRGB images, each block is 256×256 pixels. The blocks are taken from 32 images, 32 blocks from each image ($32 \times 32 = 1024$). All image blocks are combined in a single 4D array of shape $[1024, 256, 256]$ where each consecutive 32 images belong to the same image, for example, the first 32 images belong to the first image, and so on. The blocks have the same number format as the training data. Similarly, the SIDD+ testing set consists of 1024 noisy image blocks from a different set of images, but following the same format as the validation set. Image metadata files were also provided for all 64 images from which the validation and testing data were extracted. All newly created validation and testing datasets are publicly available.

Metadata and camera pipeline To facilitate the development of image denoisers targeting the raw-RGB color space, we developed a simulated camera pipeline used to render raw-RGB images into sRGB for the SIDD dataset[1]. This pipeline is provided in Python and Matlab. This pipeline is designed to be easily integrated in image denoising frameworks, especially deep learning ones. The provided pipeline offers a set of processing stages similar to an on-board camera pipeline. Such stages include: black level subtraction, active area cropping, white balance, colour space transformation, and global tone mapping. A block diagram of this pipeline is shown in Figure 7.1. Additionally, we provided tools for extracting camera metadata such as Bayer patterns and noise level functions. Such tools are helpful for designing neural network denoisers that target multiple cameras and/or multiple noise profiles. We used such tools to extract the noise level estimates of the SIDD images. The range of noise level functions (NLFs) is $[1.1841^{-4}, 2.1949^{-2}]$ for β_1 and $[2.0024^{-06}, 1.7506^{-3}]$ for β_2. For Gaussian σ, the estimated range is $[0.242, 11.507]$ in the space of $[0, 255]$.

[1]https://github.com/AbdoKamel/simple-camera-pipeline

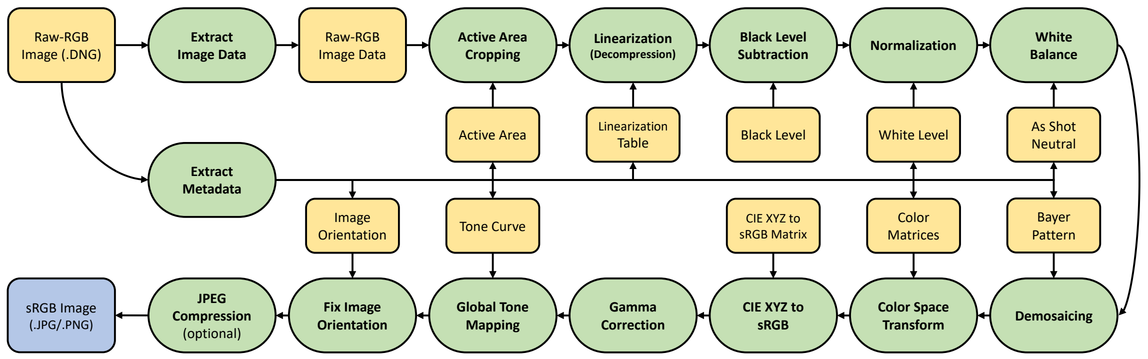


Figure 7.1: A block diagram of the simulated camera rendering pipeline provided with the NTIRE challenge.

Evaluation The evaluation is based on the comparison of the restored clean (denoised) images with the ground-truth images. For this we use the standard peak signal-to-noise ratio (PSNR) and, complementary, the structural similarity (SSIM) index [118] as often employed in the literature. Implementations are found in most of the image processing toolboxes. We report the average results over all image blocks provided.

For submitting the results, participants were asked to provide the denoised image blocks in a multidimensional array shaped in the same way as the input data. In addition, participants were asked to provide additional information: the algorithm's runtime per mega pixel (in seconds); whether the algorithm employs CPU or GPU at runtime; and whether extra metadata is used as inputs to the algorithm.

7.3 Challenge Results

In both versions of the challenge, a few hundreds of registered participants and several teams participated and submitted results, codes/executables, and factsheets. For the 2019 version of the challenge, Tables 7.1 and 7.2 report the final test results, in terms of peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) index [118], for the raw-RGB and sRGB tracks, respectively. Tables 7.3 and 7.4 show the corresponding results for the 2020 version of the challenge. The tables show the method ranks based on each measure in subscripts. We present the self-reported runtimes and major details provided in the

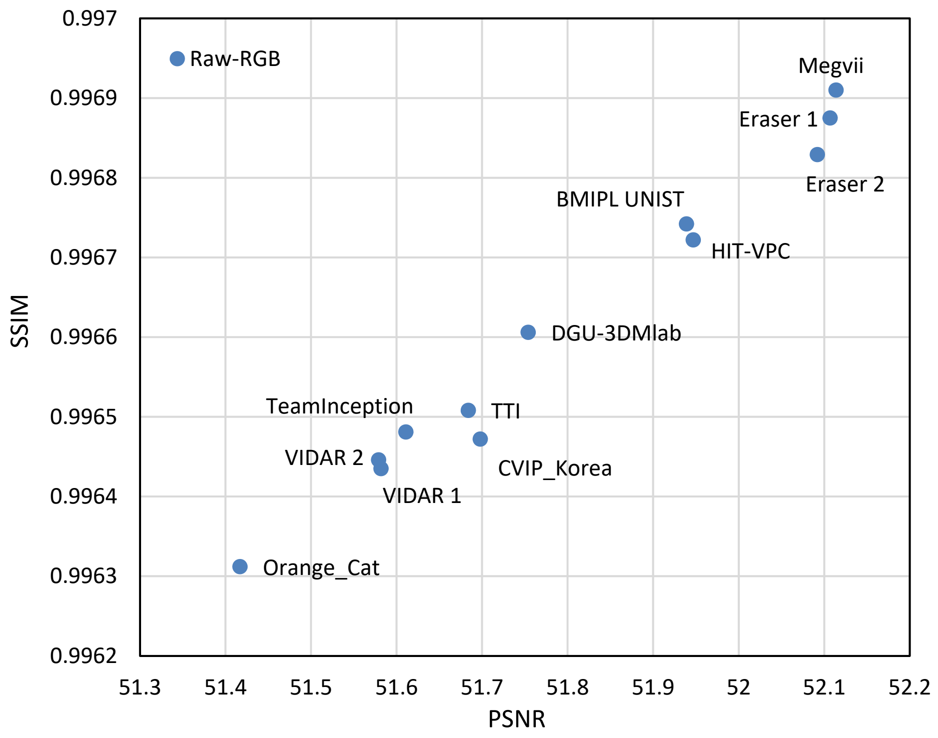


Figure 7.2: Combined PSNR and SSIM values of method from the 2019 raw-RGB track.

factsheets submitted by participants. The proposed methods are described and the teams' members are listed in the respective reports for the 2019 [4] and 2020 [1] challenges.

Figures 7.2 and 7.3 show a 2D visualization of PSNR and SSIM values for all methods in both raw-RGB and sRGB tracks, respectively, from the 2019 challenge. For combined visualization, both figures are overlaid in Figure 7.6. Figures 7.4, 7.5, 7.7 show the corresponding results from the 2020 challenge.

Main ideas All of the proposed methods participating in the challenge are based on deep learning. Specifically, all methods employ convolutional neural networks (CNNs) based on various architectures. Most of adapted architectures are based on widely-used networks, such as U-Net [108], ResNet [63], and DenseNet [68]. The main ideas included re-structuring existing networks, introducing skip connections, introducing residual connections, and using densely connected components. Other strategies have been used such as feature attention for image denoising [10], atrous spatial pyramid pooling (ASPP) [27],

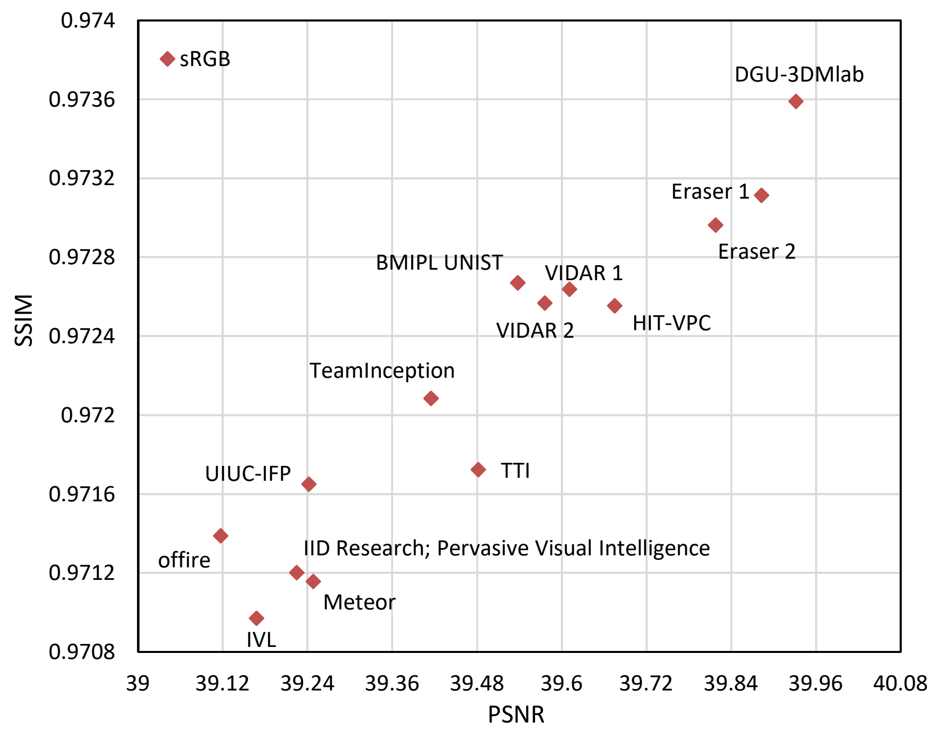


Figure 7.3: Combined PSNR and SSIM values of method from the 2019 sRGB track.

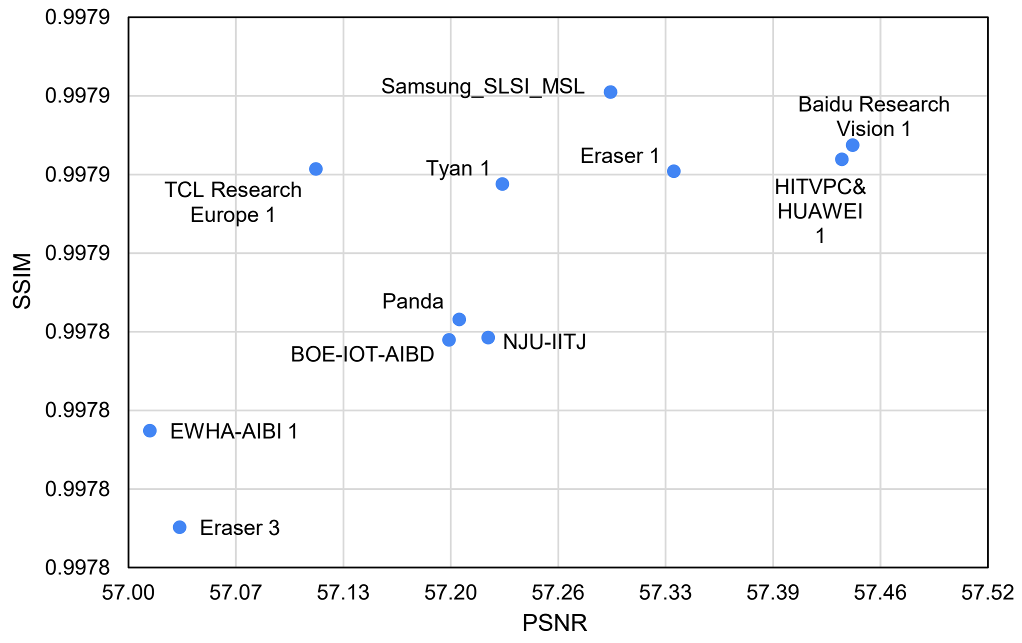


Figure 7.4: Combined PSNR and SSIM values of method from the 2020 rawRGB track.

and neural architectural search (NAS) [44].

Most teams used L_2 or L_1 loss as the optimization function while some teams adopted

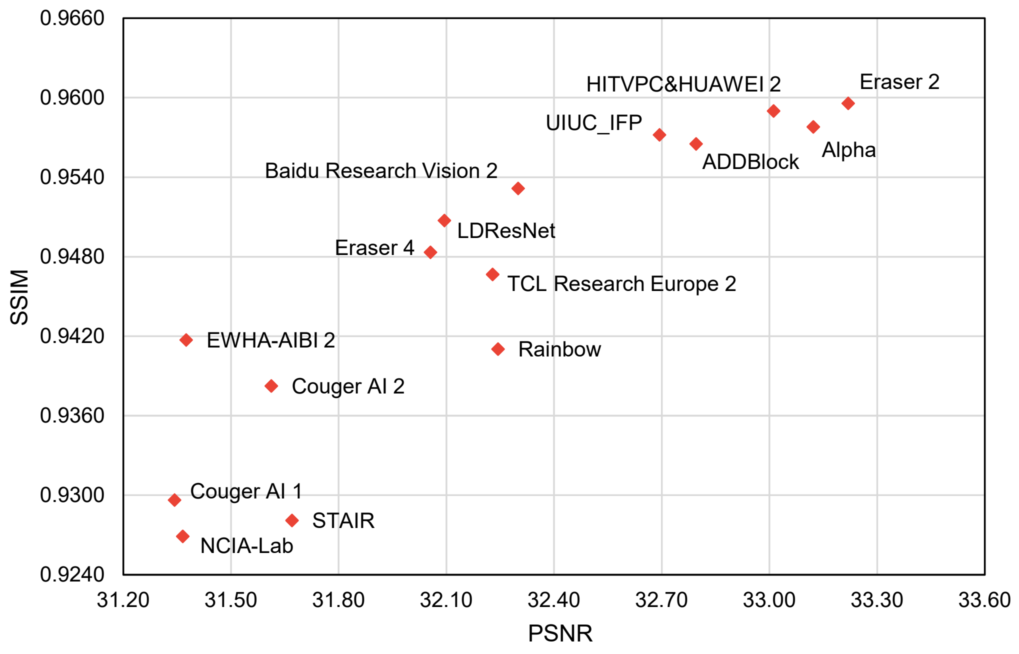


Figure 7.5: Combined PSNR and SSIM values of method from the 2020 sRGB track.

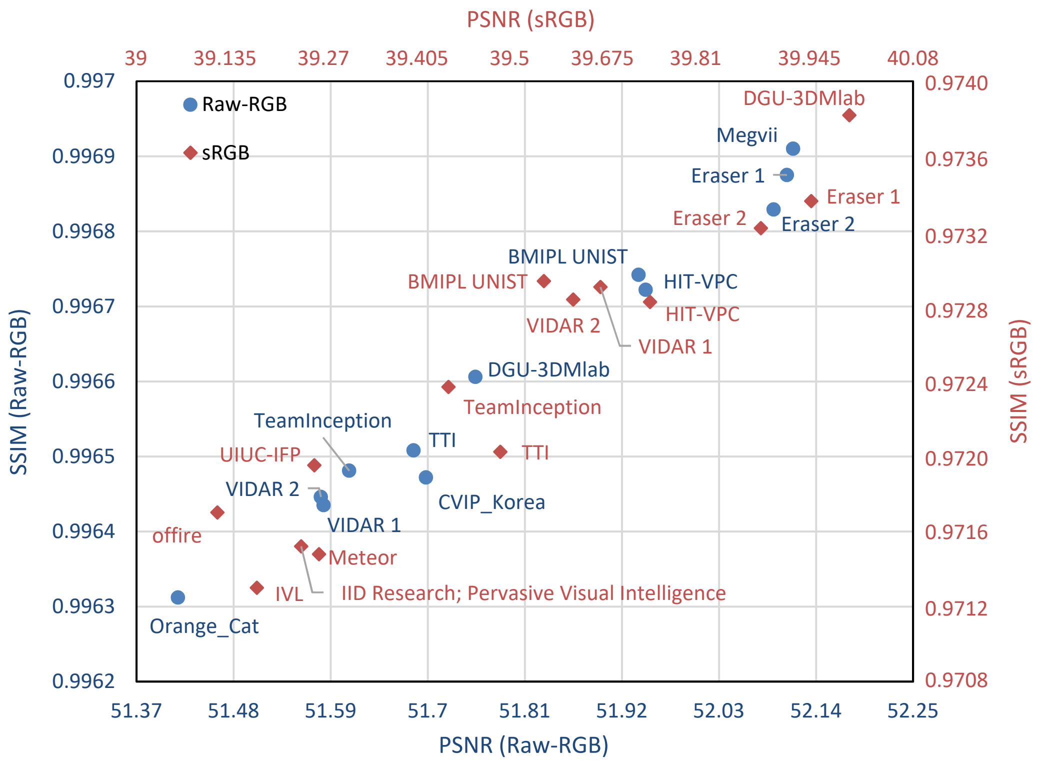


Figure 7.6: Combined PSNR and SSIM values of all methods from both raw-RGB (in blue) and sRGB (in red) tracks from the 2019 challenge. Note the different axes and scales for each track.

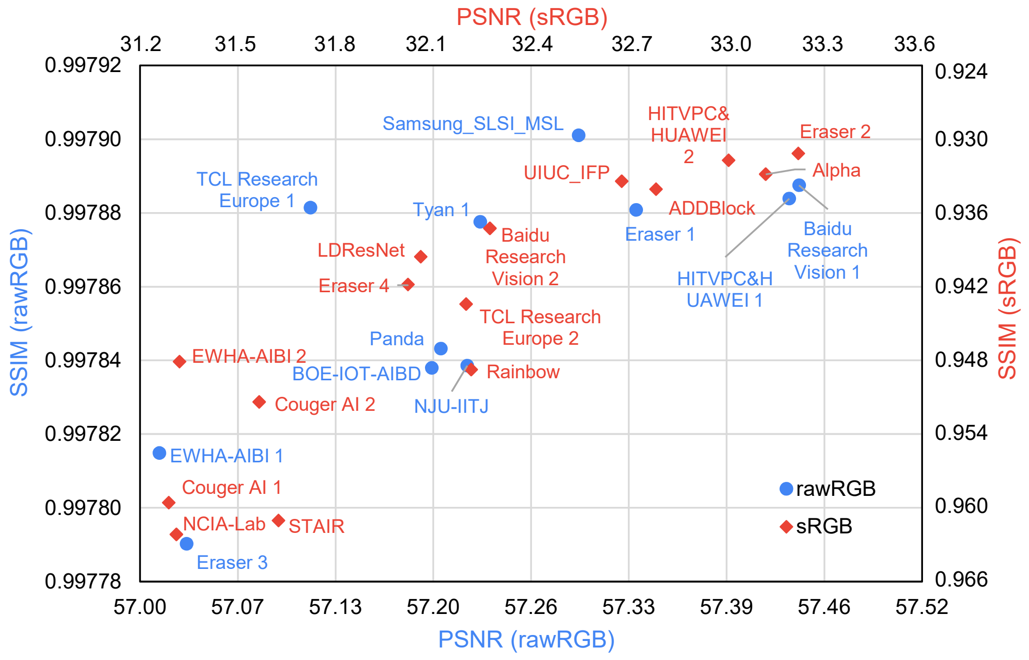


Figure 7.7: Combined PSNR and SSIM values of all methods from both rawRGB (in blue) and sRGB (in red) tracks from the 2020 challenge. Note the different axes and scales for each track.

a mixed loss between L_1 and multi-scale structural similarity (MS-SSIM) [119] and/or Laplace gradients. Some teams used KL divergence as the optimization function to infer the mean value of a pixel as well as the noise variance at that pixel's location. More details

Top results In the 2019 challenge, the top three methods achieved very close performances, in terms of PSNR and SSIM, with less than 0.02 dB differences in raw-RGB space, as shown in Figure 7.2, and with less than $0.06dB$ differences in the sRGB, as shown in Figure 7.3. The differences in SSIM values were similarly close, with less than 1×10^{-4} in raw-RGB space and less than 5×10^{-4} in sRGB space. The best performing method for raw-RGB denoising (team Megvii [83]) achieved 52.114 dB PSNR while the best method for sRGB denoising (team DGU-3DMlab [73]) achieved 39.932 dB PSNR. The next best two methods in both tracks were proposed by team Eraser [123, 108], as shown in Figure 7.6.

Team	Username	PSNR	SSIM	Runtime (s/Mpixel)	CPU/GPU (at runtime)	Platform	Ensemble	Loss
Megvii	memono11	$52.114_{(1)}$	$0.9969_{(1)}$	0.169	RTX 2080Ti	PyTorch	models ensemble ($\times 8$)	L_1
Eraser	Songsaris	$52.107_{(2)}$	$0.9969_{(2)}$	3.381	RTX 2080Ti	PyTorch	models ($\times 2$), flip/rotate ($\times 8$)	L_1
Eraser	kkbbbj	$52.092_{(3)}$	$0.9968_{(3)}$	~ 2	RTX 2080Ti	PyTorch	models ($\times 2$), flip/rotate ($\times 8$)	L_1
HIT-VPC	opt	$51.947_{(4)}$	$0.9967_{(5)}$	–	GTX 1080Ti	PyTorch	flip/rotate ($\times 8$)	L_1
BMIPL UNIST	BMIPL_denoiser	$51.939_{(5)}$	$0.9967_{(4)}$	3.132	Titan X	TensorFlow/ PyTorch	models ($\times 3$), flip/rotate ($\times 8$)	Mixed (L_1 and MS-SSIM)
DGU-3DMlab	DGU-3DMlab1	$51.754_{(6)}$	$0.9966_{(6)}$	0.8965	Titan Xp	PyTorch	None	L_1
CVIP_Korea	DP_Lim	$51.698_{(7)}$	$0.9965_{(9)}$	1.983	Titan Xp	TensorFlow	None	L_2
TTI	iim_lab	$51.684_{(8)}$	$0.9965_{(7)}$	2	Titan X	PyTorch	flip/rotate ($\times 8$)	L_1
TeamInception	swz30	$51.611_{(9)}$	$0.9965_{(8)}$	0.48	Titan Xp	PyTorch	flip/rotate ($\times 8$)	MSE
VIDAR	ChangC	$51.582_{(10)}$	$0.9964_{(11)}$	0.665	Tesla V100	PyTorch	models ($\times 3$), flip/rotate ($\times 8$)	MSE
VIDAR	eubear	$51.579_{(11)}$	$0.9964_{(10)}$	0.665	Tesla V100	PyTorch	models ($\times 3$), flip/rotate ($\times 8$)	MSE
Orange_Cat	orange_cat	$51.417_{(12)}$	$0.9963_{(12)}$	0.064	GTX 1080Ti	TensorFlow	models ($\times 11$)	L_1

Table 7.1: Results and rankings of methods submitted to the 2019 raw-RGB denoising track.

In the 2020 challenge, the top methods achieved very close performances as well, in terms of PSNR and SSIM. In the rawRGB track, the top two methods are 0.01 dB apart in terms of PSNR, whereas in the sRGB track, the top three methods have $\sim$ 0.1 dB difference in terms of PSNR, as shown in Figures 7.4 and 7.5. The differences in SSIM values were similarly close. In terms of PSNR, the main performance metric used in the challenge, the best two methods for rawRGB denoising are proposed by teams Baidu Research Vision and HITVPC&HUAWEI, and achieved 57.44 and 57.43 dB PSNR, respectively, while the best method for sRGB denoising is proposed by team Eraser and achieved 33.22 dB PSNR. In terms of SSIM, as a complementary performance metric, the best method for rawRGB denoising is proposed by the team Samsung_SLSI_MSL [12]

Team	Username	PSNR	SSIM	Runtime (s/Mpixel)	CPU/GPU (at runtime)	Platform	Ensemble	Loss
DGU-3DMlab	DGU-3DMlab	$39.932_{(1)}$	$0.9736_{(1)}$	0.5577	Titan Xp	PyTorch	None	L_1
Eraser	kkbbbj	$39.883_{(2)}$	$0.9731_{(2)}$	~ 2	GTX 1080Ti	PyTorch	models ($\times 2$), flip/rotate ($\times 8$)	L_1
Eraser	Songsaris	$39.818_{(3)}$	$0.973_{(3)}$	3.416	GTX 1080Ti	PyTorch	models ($\times 2$), flip/rotate ($\times 8$)	L_1
HIT-VPC	opt	$39.675_{(4)}$	$0.9726_{(7)}$	–	GTX 1080Ti	PyTorch	flip/rotate ($\times 8$)	L_1
VIDAR	eubear	$39.611_{(5)}$	$0.9726_{(5)}$	0.903	Tesla V100	PyTorch	models ($\times 3$), flip/rotate ($\times 8$)	MSE
VIDAR	ChangC	$39.576_{(6)}$	$0.9726_{(6)}$	0.903	Tesla V100	PyTorch	models ($\times 3$), flip/rotate ($\times 8$)	MSE
BMIPL UNIST	BMIPL_denoiser	$39.538_{(7)}$	$0.9727_{(4)}$	3.132	Titan X	TensorFlow/ PyTorch	models ($\times s$), flip/rotate ($\times 8$)	Mixed (L_1 and MS-SSIM)
TTI	iim_lab	$39.482_{(8)}$	$0.9717_{(9)}$	~ 2	Titan X	PyTorch	flip/rotate ($\times 8$)	L_1
TeamInception	swz30	$39.415_{(9)}$	$0.9721_{(8)}$	1.136	Titan Xp	PyTorch	flip/rotate ($\times 8$)	MSE
Meteor	loseall	$39.248_{(10)}$	$0.9712_{(13)}$	0.13	Titan X	TensorFlow/ PyTorch	flip/rotate ($\times 8$)	Multi-level L_1
UIUC-IFP	fyc0624	$39.242_{(11)}$	$0.9717_{(10)}$	10.73	GPU	TensorFlow	flip/rotate ($\times 8$)	–
IID Research; Pervasive Visual Intelligence	zsyue	$39.225_{(12)}$	$0.9712_{(12)}$	0.0283	RTX 2080Ti	PyTorch	flip/rotate ($\times 8$)	KL divergence
IVL	Zino	$39.168_{(13)}$	$0.971_{(14)}$	0.02	Titan V	PyTorch	model snapshots ($\times 3$ epochs)	L_1
offire	of-fire	$39.117_{(14)}$	$0.9714_{(11)}$	3.83	Titan Xp	PyTorch	None	L_1

Table 7.2: Results and rankings of methods submitted to the 2019 sRGB denoising track.

and achieved a SSIM index of 0.9979, while the best SSIM index for sRGB denoising is achieved by the Eraser team.

Ensembles To boost performance, most of the methods applied different flavours of ensemble techniques. Specifically, most teams used a self-ensemble [116] technique where the results from eight flipped/rotated versions of the same image are averaged together. Some teams applied additional model-ensemble techniques.

Team	Username	PSNR	SSIM	Runtime (s/Mpixel)	CPU/GPU (at runtime)	Platform	Ensemble	Loss
Baidu Research Vision 1	zhihongp	$57.44_{(1)}$	$0.99789_{(2)}$	5.76	Tesla V100	PaddlePaddle, PyTorch	flip/transpose (×8)	L_1
HITVPC&HUAWEI 1	hitvpc_huawei	$57.43_{(2)}$	$0.99788_{(3)}$	–	GTX 1080 Ti	PyTorch	flip/rotate (×8)	L_1
Eraser 1	Songsaris	$57.33_{(3)}$	$0.99788_{(5)}$	36.50	TITAN V	PyTorch	flip/rotate (×8)	L_1
Samsung_SLSI_MSL	Samsung_SLSI_MSL-2	$57.29_{(4)}$	$0.99790_{(1)}$	50	Tesla V100	PyTorch	flip/transpose (×8), models (×3)	L_1
Tyan 1	Tyan	$57.23_{(5)}$	$0.99788_{(6)}$	0.38	GTX 1080 Ti	TensorFlow	flip/rotate (×8), model snapshots (×3)	L_1
NJU-IITJ	Sora	$57.22_{(6)}$	$0.99784_{(9)}$	3.5	Tesla V100	PyTorch	models (×8)	L_1
Panda	panda_ynn	$57.20_{(7)}$	$0.99784_{(8)}$	2.72	GTX 2080 Ti	TensorFlow	flip/rotate (×8), model snapshots (×3)	L_1
BOE-IOT-AIBD	eastworld	$57.19_{(8)}$	$0.99784_{(7)}$	0.61	Tesla P100	TensorFlow	None	L_1
TCL Research Europe 1	tcl-research-team	$57.11_{(9)}$	$0.99788_{(10)}$	–	RTX 2080 Ti	TensorFlow	flip/rotate (×8), models (×3−5)	L_1
Eraser 3	BumjunPark	$57.03_{(10)}$	$0.99779_{(4)}$	0.31	–	PyTorch	–	L_1
EWHA-AIBI 1	jaayeon	$57.01_{(11)}$	$0.99781_{(12)}$	55	Tesla V100	PyTorch	flip/rotate (×8)	L_1
ZJU231	qiushizai	$56.72_{(12)}$	$0.99752_{(11)}$	0.17	GTX 1080 Ti	PyTorch	self ensemble	L_1, DCT
NoahDn	matteomaggioni	$56.47_{(13)}$	$0.99749_{(14)}$	3.54	Tesla V100	TensorFlow	flip/rotate (×8)	L_1
Dahua_isp	–	$56.20_{(14)}$	$0.99749_{(13)}$	–	GTX 2080	PyTorch	–	–

Table 7.3: Results and rankings of methods submitted to the 2020 rawRGB denoising track.

Training data Most of the teams relied solely on the training data provided by the SIDD dataset while applying usual data augmentation strategies, such as flipping and rotating images. However, some teams (e.g., Meteor) used additional training data from other datasets, such as DIV2K dataset [115, 6], DSB500 dataset [11], and Waterloo Exploration Database [89].

Results conclusion From the analysis of the presented results, we can conclude that the proposed methods achieve state-of-the-art performance in real image denoising on the SIDD and SIDD+ benchmarks. The methods proposed by the top ranking teams in each challenge achieve consistent performance across both colour spaces (see Figures 7.6 and 7.7).

Team	Username	PSNR	SSIM	Runtime (s/Mpixel)	CPU/GPU (at runtime)	Platform	Ensemble	Loss
Eraser 2	Songsaris	$33.22_{(1)}$	$0.9596_{(1)}$	103.92	TITAN V	PyTorch	flip/rotate/RGB shuffle ($\times 48$)	L_1
Alpha	q935970314	$33.12_{(2)}$	$0.9578_{(3)}$	6.72	RTX 2080 Ti	PyTorch	flip/rotate ($\times 8$)	Charbonnier
HITVPC&HUAWEI 2	hitvpc_huawei	$33.01_{(3)}$	$0.9590_{(2)}$	–	GTX 1080 Ti	PyTorch	flip/rotate ($\times 8$)	L_1
ADDBlock	BONG	$32.80_{(4)}$	$0.9565_{(5)}$	76.80	Titan XP	PyTorch	flip/rotate ($\times 8$), models ($\times 4$)	L_1
UIUC_IFP	Self-Worker	$32.69_{(5)}$	$0.9572_{(4)}$	0.61	Tesla V100 ($\times 2$)	PyTorch	flip/rotate ($\times 8$), models($\times 3$)	L_1
Baidu Research Vision 2	zhihongp	$32.30_{(6)}$	$0.9532_{(6)}$	9.28	Tesla V100 ($\times 8$)	PaddlePaddle, PyTorch	flip/transpose ($\times 8$)	L_1
Rainbow	JiKun63	$32.24_{(7)}$	$0.9410_{(11)}$	2.41	RTX 2080Ti	PyTorch	flip/rotate ($\times 8$)	L_1/Laplace gradient
TCL Research Europe 2	tcl-research-team	$32.23_{(8)}$	$0.9467_{(9)}$	–	RTX 2080 Ti	TensorFlow	flip/rotate ($\times 8$), models ($\times 3-5$)	L_1
LDResNet	SJKim	$32.09_{(9)}$	$0.9507_{(7)}$	17.85	GTX 1080	PyTorch	flip/rotate ($\times 8$)	L_1
Eraser 4	BumjunPark	$32.06_{(10)}$	$0.9484_{(8)}$	–	–	PyTorch	–	L_1
STAIR	dark_1im1ess	$31.67_{(11)}$	$0.9281_{(14)}$	1.86	Titan TITAN RTX ($\times 2$)	–	–	L_1
Couger AI 2	priyakansal	$31.61_{(12)}$	$0.9383_{(12)}$	0.23	GTX 1080	Keras/Tensorflow	None	MSE/SSIM
EWHA-AIBI 2	jaayeon	$31.38_{(13)}$	$0.9417_{(10)}$	–	Tesla V100	PyTorch	flip/rotate ($\times 8$)	L_1
NCIA-Lab	Han-Soo-Choi	$31.37_{(14)}$	$0.9269_{(15)}$	2.92	TITAN RTX	PyTorch	None	MS-SSIM/$L1$
Couger AI 1	sabarinathan	$31.34_{(15)}$	$0.9296_{(13)}$	0.23	GTX 1080	Keras/Tensorflow	None	MSE/SSIM
Visionaries	rajatguptakgp	$19.97_{(16)}$	$0.6791_{(16)}$	–	GTX 1050 Ti	PyTorch	None	MSE

Table 7.4: Results and rankings of methods submitted to the 2020 sRGB denoising track.

7.4 Conclusion

In this chapter, we discussed our contribution towards providing the research community with a comprehensive framework for image denoising evaluation. Our contribution was realised as the NTIRE image denoising challenge. In this challenge, we provide two challenge tracks for benchmarking image denoisers in both raw-RGB and sRGB colour spaces. To facilitate the development of image denoisers in the raw-RGB color space, we developed important tools including a simulated camera rendering pipeline and camera metadata extraction tools. We believe our image denoising challenge and framework would be helpful to researchers in the area of image denoising.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

This dissertation has presented a comprehensive review on the research areas of image noise modelling, estimation, and reduction in Chapters 2, 3, and 4. The dissertation also discussed a set of research directions and presented a number of interesting contributions to the research field of image noise modelling and reduction. The contributions includes datasets, models, source codes, and interesting findings that are believed to be valuable in advancing the state of the art in the respective research filed. These contributions are summarized in the following.

First, in Chapter 5, we proposed a systematic procedure for estimating ground truth for real noisy images that can be used to benchmark denoising performance on smartphone imagery. Using this procedure, we captured a the Smartphone Image Denoising Dataset (SIDD) of $\sim 30,000$ real noisy images using five representative smartphone cameras and generated their ground truth images. Using our dataset, we benchmark a number of denoising methods to gauge the relative performance of various approaches, including patch-based methods and more recent CNN-based techniques. From the provided analysis, we show that for CNN-based methods, notable gains can be made when using our ground truth data versus conventional alternatives such as low-ISO images. The SIDD has many merits including high quality, representing multiple cameras and multiple image capture settings, being large-scale and more suitable for training learning-based methods. However, it requires a special image capturing setup; and it does not contain outdoor images or daylight images.

Second, in Chapter 6, using the aforementioned dataset, we developed a novel noise model (Noise Flow) that combines the insights of parametric noise models and the expressiveness of powerful normalising flow models. We leveraged recent normalizing flow architectures to accurately model noise distributions observed from large datasets of real noisy images. In particular, based on the recent Glow architecture, we constructed a normalising flow model that is conditioned on critical camera variables, such as intensity, camera type, and gain settings. The model was shown to be a strict generalization of the camera-specific noise models but with the ability to capture significantly more complex behaviour. The result is a single model that is compact (fewer then 2500 parameters) and considerably more accurate than existing models. To demonstrate the effectiveness of Noise Flow, we used it to synthesize training data for image denoising resulting in significant improvements in image quality. Noise flow is accurate, realistic, and lightweight; however, it requires many images to train, and it is trained on rawRGB image only.

Lastly, in Chapter 7, using our SIDD dataset, we provided the research community with a comprehensive framework for image denoising evaluation. The main part of this framework is an image denoising challenge that aims to gauge and advance the state-of-the-art in image denoising with more focus on *real*, rather than synthetic, noisy images. In this challenge, we designed two tracks for benchmarking image denoisers in both raw sensor data (raw-RGB) and standard RGB (sRGB) colour spaces. To facilitate the development of image denoisers in the raw-RGB color space, we developed important tools, including a simulated camera rendering pipeline and camera metadata extraction tools. This challenge has been hosted twice in the workshop on New Trends in Image

Restoration and Enhancement (NTIRE), in 2019 and 2020. This benchmark is the first public benchmark that targets quantitative evaluation on real images from smartphone cameras. However, we still think the aggregate evaluation metrics are not sufficient, and still there is no visual or perceptual assessment of results, and no restrictions on runtime or model size.

We believe our study including the developed methods, datasets, noise models, and benchmarks will be useful in advancing image denoising and other computer vision tasks for images captured with smartphones. All developed methods and source codes have been made publicly available to the research community (links are in Appendix 8.2.3).

8.2 Future Research Directions

Following up on the many observations made in Chapter 4 about the image denoising strategies and common practices, it is clear that research on image noise modelling and reduction is still lacking in many aspects. We have addressed a few of these aspects in Chapters 5, 6, and 7. The most obvious next steps in our research is to address the limitation of our proposed methods. For the ground truth image estimation, we may think of ways to extend our method to capturing natural and outdoor scene. For noise flow, we can pursue other ways of training such models that may not require many images or may not require ground truth at all. For the SIDD benchmark, we may consider going beyond the aggregate evaluation metrics and include visual or perceptual assessment of results. Also, we may consider adding more restrictions on runtime and model size. Additional research aspects and further research directions are discussed next.

8.2.1 Realistic Noise Models in Perceptual Colour Spaces

One can argue that collecting large-scale image datesets is time-consuming and requires extensive efforts. Despite the fact that computer vision research heavily depend on large-scale image datasets (e.g., the ImageNet dataset [34]), there are other directions trying to avoid depending on such datasets and to use only synthetically generated data. This direction is clearly observable in image denoising research and we have followed this direction in developing our Noise Flow generative model. However, Noise Flow targets rawRGB images, which are minimally processed and do not suffer from the complex ISP processing stages. On the other hand, it is more common for computer vision tasks to use images in perceptual colour space, such as sRGB images saved in common JPEG or PNG formats. Such images suffer from many complex transformations applied in the image processing pipeline on-board ISPs, and hence, their noise distributions can be far more complex than the minimally-processed rawRGB images.

To address the above issue, a promising research direction is to improve our Noise Flow model to be able to model noise distributions from images in perceptual colour spaces. For a Noise Flow to model such complex noise, it needs to account for all noise types occurring on-board the image processing pipeline as well. This may be achieved be carefully re-designing the Noise Flow architecture to incorporate parameters or layers that represent noise transformations in different stages of the imaging pipeline.

8.2.2 Towards Accurate Noise Estimation

Image noise estimation is tightly coupled with the assumption of the underlying noise model. For example, a noise estimation algorithm assuming an additive white Gaussian noise would be aiming at accurate estimation of the standard deviation of the noise Gaussian distribution. Analogously, another noise estimation algorithm assuming a heteroscedastic Gaussian noise model would try to estimate the parameters β_1 and β_2 from Equation 3.8. Our approach towards realistic noise estimation would be highly dependent on the development of more realistic noise models, as discussed earlier.

Existing image denoising methods can still benefit from accurate noise estimation methods; however, existing noise estimation methods are focused solely on the noise model and the input image; however, other factors can affect the noise distribution in the capture image, and subsequently, the estimation of such distribution. These factors may include the type of imaging device and the lighting environment. To this end, potential research directions towards noise estimation may include the investigation of such factors and their effects on the noise estimation process.

8.2.3 Towards Better Image Denoising

Image denoising is basically the application of an algorithm that processes a noisy image, assuming a specific noise model, and optionally having access to the estimated noise model parameters (i.e., the noise level) for the noisy image, with the outcome of having a noise-free version of that noisy image. To improve image denoising, we can improve any of the aforementioned ingredients. We may choose to improve the accuracy of the assumed noise model in representing the noisy images; improve the accuracy of how the model parameters are estimated; or improve the image denoising algorithm itself. However, there are many aspects that affect all of these factors that also need to be taken into account, such as the imaging device and the imaging environment. To improve image denoising, these aspects need to be factored in the noise models and the denoising algorithms as well.

A promising research direction towards improving image denoising is to provide a framework to harness all of the aforementioned ingredients and factors into the denoising process. Another research direction that have been receiving more attention recently is to focus more on specific cases, such as denoising in low-light environments [86, 61, 91] or denoising that targets specific types of images or imaging devices [51, 105, 106, 76]. We hope to investigate some of these research directions in the near future.

Bibliography

[1] Abdelrahman Abdelhamed, Mahmoud Afifi, Radu Timofte, Michael S. Brown, Yue Cao, Zhilu Zhang, Wangmeng Zuo, Xiaoling Zhang, Jiye Liu, Wendong Chen, Changyuan Wen, Meng Liu, Shuailin Lv, Yunchao Zhang, Zhihong Pan, Baopu Li, Teng Xi, Yanwen Fan, Xiyu Yu, Gang Zhang, Jingtuo Liu, Junyu Han, Errui Ding, Songhyun Yu, Bumjun Park, Jechang Jeong, Shuai Liu, Ziyao Zong, Nan Nan, Chenghua Li, Zengli Yang, Long Bao, Shuangquan Wang, Dongwoon Bai, Jungwon Lee, Youngjung Kim, Kyeongha Rho, Changyeop Shin, Sungho Kim, Pengliang Tang, Yiyun Zhao, Yuqian Zhou, Yuchen Fan, Thomas Huang, Zhihao Li, Nisarg A. Shah, Wei Liu, Qiong Yan, Yuzhi Zhao, Marcin Mȯzejko, Tomasz Latkowski, Lukasz Treszczotko, Michael Szafraniuk, Krzysztof Trojanowski, Yanhong Wu, Pablo Navarrete Michelini, Fengshuo Hu, Yunhua Lu, Sujin Kim, Wonjin Kim, Jaayeon Lee, Jang-Hwan Choi, Magauiya Zhussip, Azamat Khassenov, Jong Hyun Kim, Hwechul Cho, Priya Kansal, Sabari Nathan, Ye Zhangyu, Lu Xiwen, Wu Yaqi, Yang Jiangxin, Cao Yanlong, Tang Siliang, Cao Yanpeng, Matteo Maggioni, Ioannis Marras, Thomas Tanay, Gregory Slabaugh, Youliang Yan, Myungjoo Kang, Han-Soo Choi, Kyungmin Song, Shusong Xu, Xiaomu Lu, Tingniao Wang, Chunxia Lei, Bin Liu, Rajat Gupta, and Vineet Kumar. NTIRE 2020 challenge on real image denoising: Dataset, methods and results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 496–497, June 2020. 99

[2] Abdelrahman Abdelhamed, Marcus A Brubaker, and Michael S Brown. Noise flow: Noise modeling with conditional normalizing flows. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3165–3173, October 2019. 86

[3] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. A high-quality denoising dataset for smartphone cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1692–1700, June 2018. xv, 18, 35, 39, 70, 71, 73, 76, 77, 85, 94

[4] Abdelrahman Abdelhamed, Radu Timofte, Michael S. Brown, Songhyun Yu, Bumjun Park, Jechang Jeong, Seung-Won Jung, Dong-Wook Kim, Jae-Ryun Chung, Jiaming Liu, Yuzhi Wang, Chi-Hao Wu, Qin Xu, Yuqian Zhou, Chuan Wang, Shaofan Cai, Yifan Ding, Haoqiang Fan, Jue Wang, Kai Zhang, Wangmeng Zuo, Magauiya Zhussip, Dong Won Park, Shakarim Soltanayev, Se Young Chun, Zhiwei Xiong, Chang Chen, Muhammad Haris, Kazutoshi Akita, Tomoki Yoshida, Greg Shakhnarovich, Norimichi Ukita, Syed Waqas Zamir, Aditya Arora, Salman Khan, Fahad Shahbaz Khan, Ling Shao, Sung-Jea Ko, Dong-Pan Lim, Seung-Wook Kim, Seo-Won Ji, Sang-Won Lee, Wenyi Tang, Yuchen Fan, Yuqian Zhou, Ding Liu, Thomas S. Huang, Deyu Meng, Lei Zhang, Hongwei Yong, Yiyun Zhao, Pengliang Tang, Yue Lu, Raimondo Schettini, Simone Bianco, Simone Zini, Chi Li, Yang Wang, and Zhiguo Cao. NTIRE 2019 challenge on real image denoising: Methods and results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019. 99

[5] Mahmoud Afifi, Abdelrahman Abdelhamed, Abdullah Abuolaim, Abhijith Punnappurath, and Michael S Brown. CIE XYZ Net: Unprocessing images for low-level computer vision tasks. *arXiv preprint arXiv:2006.12709*, 2020. 39

[6] Eirikur Agustsson and Radu Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017. 105

[7] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006. 62, 63

[8] Miika Aittala and Frédo Durand. Burst Image Deblurring Using Permutation Invariant Convolutional Neural Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 86

[9] Josue Anaya and Adrian Barbu. Renoir–a dataset for real low-light image noise reduction. *Journal of Visual Communication and Image Representation*, 51:144–154, 2018. 40, 41, 44

[10] Saeed Anwar and Nick Barnes. Real image denoising with feature attention. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3155–3164, 2019. 99

[11] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(5):898–916, May 2011. 105

[12] Long Bao, Zengli Yang, Shuangquan Wang, Dongwoon Bai, and Jungwon Lee. Real image denoising based on multi-scale residual dense block and cascaded U-Net with block-connection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020. 103

[13] Marcelo Bertalmío. *Denoising of photographic images and video: fundamentals, open challenges and new trends.* Springer, 2018. 23

[14] Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 11(6):567–585, 1989. 54

[15] Alan C Bovik. *The essential guide to image processing*. Academic Press, 2009. 3, 23

[16] Tim Brooks, Ben Mildenhall, Tianfan Xue, Jiawen Chen, Dillon Sharlet, and Jonathan T Barron. Unprocessing images for learned raw denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 39

[17] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 60–65, 2005. 30

[18] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation*, 4(2):490–530, 2005. 23

[19] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. Non-local means denoising. *Image Processing On Line*, 1:208–212, 2011. 30

[20] Antoni Buades, Bartomeu Coll, and JM Morel. A non-local algorithm for image denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 62, 63, 93

[21] Antoni Buades, Yifei Lou, Jean-Michel Morel, and Zhongwei Tang. Multi image noise estimation and denoising. *MAP5 2010-19*, 2010. 35, 42

[22] HC. Burger, CJ. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with BM3D? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 32, 62, 63

[23] Priyam Chatterjee and Peyman Milanfar. Is denoising dead? *IEEE Transactions on Image Processing (TIP)*, 19(4):895–911, 2010. 32

[24] Priyam Chatterjee and Peyman Milanfar. Patch-based near-optimal image denoising. *IEEE Transactions on Image Processing (TIP)*, 21(4):1635–1649, 2012. 31

[25] Guangyong Chen, Fengyuan Zhu, and Pheng Ann Heng. An efficient statistical method for image noise level estimation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015. 16, 35, 50, 60

[26] Jingwen Chen, Jiawei Chen, Hongyang Chao, and Ming Yang. Image Blind Denoising with Generative Adversarial Network Based Noise Modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 73

[27] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 801–818, 2018. 99

[28] Yunjin Chen and Thomas Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(6):1256–1272, 2017. 62, 63

[29] A Clifford Cohen. *Truncated and censored samples: theory and applications*. CRC Press, 2016. 19

[30] Dalsa Corporation. Electronic shuttering for high speed CMOS machine vision applications. `https://www.velocimetry.net/downloads/Photonik_CMOS_Shuttering_English.pdf`. Accessed: 2019-01-21. 10

[31] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising with block-matching and 3d filtering. In *Image Processing: Algorithms*

and Systems, Neural Networks, and Machine Learning, volume 6064, page 606414. International Society for Optics and Photonics, 2006. 31

[32] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3D transform-domain collaborative filtering. *IEEE Transactions on Image Processing (TIP)*, 16(8):2080–2095, 2007. xii, 31, 32, 33, 62, 63, 93

[33] Antonio De Stefano, Paul R White, and William B Collis. Training methods for image noise level estimation on wavelet components. *EURASIP Journal on Applied Signal Processing*, 2004:2400–2407, 2004. 35

[34] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009. 111

[35] Steven Diamond, Vincent Sitzmann, Stephen Boyd, Gordon Wetzstein, and Felix Heide. Dirty Pixels: Optimizing Image Classification Architectures for Raw Sensor Data. *arXiv preprint arXiv:1701.06487*, 2017. 86

[36] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear Independent Components Estimation. In *Proceedings of the International Conference on Learning Representations (ICLR) Workshop*, 2015. 69, 73, 75

[37] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density Estimation using Real NVP. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. 75

[38] David L Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995. 35

[39] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1033–1038, 1999. 30

[40] Munir El-Desouki, M Jamal Deen, Qiyin Fang, Louis Liu, Frances Tse, and David Armstrong. CMOS image sensors for high speed applications. *Sensors*, 9(1):430–444, 2009. 10

[41] Michael Elad. On the origin of the bilateral filter and ways to improve it. *IEEE Transactions on Image Processing (TIP)*, 11(10):1141–1151, 2002. 28

[42] Michael Elad. *Sparse and redundant representations: from theory to applications in signal and image processing*. Springer Science & Business Media, 2010. 23

[43] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing (TIP)*, 15(12):3736–3745, 2006. 62, 63

[44] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*, 2018. 100

[45] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *International Journal on Computer Vision (IJCV)*, 88(2):303–338, 2010. 40

[46] Alessandro Foi. Clipped noisy images: Heteroskedastic modeling and practical denoising. *Signal Processing*, 89(12):2609–2629, 2009. 14, 18, 19, 42, 58, 70, 72, 73

[47] Alessandro Foi, Mejdi Trimeche, Vladimir Katkovnik, and Karen Egiazarian. Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data. *IEEE Transactions on Image Processing (TIP)*, 17(10):1737–1754, 2008. 3, 18, 36, 48, 50

[48] Alessandro Foi, Mejdi Trimeche, Vladimir Katkovnik, and Karen Egiazarian. Practical Poissonian-Gaussian Noise Modeling and Fitting for Single-Image Raw-Data. *IEEE Transactions on Image Processing (TIP)*, 17(10):1737–1754, 2015. 70, 72

[49] Eric R Fossum. CMOS image sensors: Electronic camera-on-a-chip. *IEEE Transactions on Electron Devices*, 44(10):1689–1698, 1997. 8, 14

[50] Gartner, Inc. Worldwide sales of smartphones. Online, February 2017. Retrieved March 9, 2018 from https://www.gartner.com/newsroom/id/3609817. 43

[51] Lovedeep Gondara. Medical image denoising using convolutional denoising autoencoders. In *Proceedings of the International Conference on Data Mining Workshops (ICDMW)*, pages 241–246. IEEE, 2016. 112

[52] Google. Android Camera2 API. https://developer.android.com/reference/android/hardware/camera2/package-summary.html. Accessed: 2017-11-10. 50

[53] Ryan D Gow, David Renshaw, Keith Findlater, Lindsay Grant, Stuart J McLeod, John Hart, and Robert L Nicol. A comprehensive tool for modeling CMOS image-sensor-noise performance. *IEEE Transactions on Electron Devices*, 54(6):1321–1329, 2007. xi, 9

[54] Glenn A Gray and Gene W Zeoli. Quantization and saturation noise due to analog-to-digital conversion. *IEEE Transactions on Aerospace and Electronic Systems*, AES-7(1):222–223, 1971. 12

[55] Robert M. Gray and David L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, 1998. 12

[56] Shuhang Gu and Radu Timofte. A brief review of image denoising algorithms and beyond. In *Inpainting and Denoising Challenges*, pages 1–21. Springer, 2019. 23, 93

[57] Shuhang Gu, Lei Zhang, Wangmeng Zuo, and Xiangchu Feng. Weighted nuclear norm minimization with application to image denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 62, 63

[58] Manuel Guizar-Sicairos, Samuel T Thurman, and James R Fienup. Efficient subpixel image registration algorithms. *Optics Letters*, 33(2):156–158, 2008. 54

[59] Samuel W Hasinoff. Photon, Poisson noise. In *Computer Vision*, pages 608–610. Springer, 2014. 3, 8

[60] Samuel W Hasinoff, Frédo Durand, and William T Freeman. Noise-optimal capture for high dynamic range photography. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. xi, 9, 12

[61] 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. *ACM Transactions on Graphics (ToG)*, 35(6), 2016. 112

[62] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(6):1397–1409, 2013. 29

[63] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 32, 75, 99

[64] Glenn Healey and Raghava Kondepudy. Radiometric CCD camera calibration and noise estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 16(3):267–276, 1994. xi, 9, 35

[65] Keigo Hirakawa. Cross-talk explained. In *Proceedings of the International Conference on Image Processing (ICIP)*, pages 677–680. IEEE, 2008. 14

[66] Gerald C Holst. *CCD arrays, cameras, and displays*. SPIE Optical Engineering Press, 1998. 11, 70

[67] Gerald C Holst and Terrence S Lomheim. *CMOS/CCD Sensors and Camera Systems*. SPIE Optical Engineering Press, 2011. 11

[68] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708, 2017. 99

[69] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 448–456, 2015. 32

[70] Xiaodan Jin and Keigo Hirakawa. Approximations to Camera Sensor Noise. In *Image Processing: Algorithms and Systems XI*, 2013. 70

[71] Hakki Karaimer and Michael S. Brown. A software platform for manipulating the camera imaging pipeline. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 63, 95

[72] Walt Kester. ADC input noise: the good, the bad, and the ugly. is no noise good noise? *Analog Dialogue*, 40(02):1–5, 2006. xi, 12, 14

[73] Dong-Wook Kim, Jae Ryun Chung, and Seung-Won and Jung. GRDN: Grouped residual dense network for real image denoising and GAN-based real-world noise modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019. 102

[74] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. 81

[75] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10236–10245, 2018. xv, 5, 69, 70, 73, 74, 75, 76, 77

[76] Pawel Korus and Nasir Memon. Neural imaging pipelines-the scourge or hope of forensics? *arXiv preprint arXiv:1902.10707*, 2019. 112

[77] Darwin T Kuan, Alexander A Sawchuk, Timothy C Strand, and Pierre Chavel. Adaptive noise smoothing filter for images with signal-dependent noise. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, PAMI-7(2):165–177, 1985. 93

[78] Jeffrey C Lagarias, James A Reeds, Margaret H Wright, and Paul E Wright. Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on Optimization*, 9(1):112–147, 1998. 59

[79] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2Noise: Learning image restoration without clean data. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2971–2980, 2018. 16, 33

[80] Jae S Lim. Two-dimensional signal and image processing. *Englewood Cliffs, NJ, Prentice Hall, 1990, 710 p.*, 1990. 25

[81] Ce Liu, William T Freeman, Richard Szeliski, and Sing Bing Kang. Noise estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 901–908. IEEE, 2006. 3, 36

[82] Ce Liu, Richard Szeliski, Sing Bing Kang, C Lawrence Zitnick, and William T Freeman. Automatic estimation and removal of noise from a single image. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(2):299–314, 2008. xi, xii, 3, 9, 18, 36, 37, 42, 48, 93

[83] Jiaming Liu, Chi-Hao Wu, Yuzhi Wang, Qin Xu, Yuqian Zhou, Haibin Huang, Chuan Wang, Shaofan Cai, Yifan Ding, Haoqiang Fan, and Jue Wang. Learning

raw image denoising with bayer pattern normalization and bayer preserving augmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019. 102

[84] Xinhao Liu, Masayuki Tanaka, and Masatoshi Okutomi. Single-image noise level estimation for blind denoising. *IEEE Transactions on Image Processing (TIP)*, 22(12):5226–5237, 2013. 35

[85] Xinhao Liu, Masayuki Tanaka, and Masatoshi Okutomi. Practical signal-dependent noise parameter estimation from a single noisy image. *IEEE Transactions on Image Processing (TIP)*, 23(10):4361–4371, 2014. xii, 18, 37, 38, 42, 72

[86] Ziwei Liu, Lu Yuan, Xiaoou Tang, Matt Uyttendaele, and Jian Sun. Fast burst images denoising. *ACM ACM Transactions on Graphics (ToG)*, 33(6):232, 2014. 112

[87] Artur Łoza, Lyudmila Mihaylova, David Bull, and Nishan Canagarajah. Structural Similarity-based Object Tracking in Multimodality Surveillance Videos. *Machine Vision and Applications*, 20(2):71–83, 2009. 85

[88] Florian Luisier, Thierry Blu, and Michael Unser. Image denoising in mixed Poisson-Gaussian noise. *IEEE Transactions on Image Processing (TIP)*, 20(3):696–708, 2011. 18

[89] 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. *IEEE Transactions on Image Processing (TIP)*, 26(2):1004–1016, Feb. 2017. 105

[90] Markku Makitalo and Alessandro Foi. Optimal inversion of the generalized Anscombe transformation for Poisson-Gaussian noise. *IEEE Transactions on Image Processing (TIP)*, 22(1):91–103, 2013. 18, 70, 72

[91] Talmaj Marinč, Vignesh Srinivasan, Serhan Gül, Cornelius Hellge, and Wojciech Samek. Multi-kernel prediction networks for denoising of burst images. *arXiv preprint arXiv:1902.05392*, 2019. 112

[92] David Martin, Charless Fowlkes, Doron Tal, Jitendra Malik, et al. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2001. 36, 40

[93] AM Mohsen, Michael F Tompsett, and Carlo H Sequin. Noise measurements in charge-coupled devices. *IEEE Transactions on Electron Devices*, 22(5):209–218, 1975. 18, 72

[94] Seonghyeon Nam, Youngbae Hwang, Yasuyuki Matsushita, and Seon Joo Kim. A holistic approach to cross-channel image noise modeling and its application to image denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. xii, 20, 39, 40, 42, 80

[95] Rang MH Nguyen and Michael S Brown. Raw image reconstruction using a self-contained sRGB-JPEG image with only 64 KB overhead. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1655–1663, 2016. 39

[96] Rang MH Nguyen and Michael S Brown. Raw image reconstruction using a self-contained sRGB-JPEG image with small memory overhead. *International Journal of Computer Vision (IJCV)*, 126(6):637–650, 2018. 39

[97] Junji Ohtsubo and Akifumi Ogiwara. Effects of clipping threshold of clipped speckle intensity. *Optics Communications*, 65(2):73–78, 1988. 14

[98] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 12(7):629–639, 1990. 27, 28

[99] Tobias Plötz and Stefan Roth. Benchmarking denoising algorithms with real photographs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. xiii, xiv, 18, 39, 40, 41, 42, 44, 45, 60, 61, 62, 64, 65, 66, 70, 73, 94

[100] Nikolay Ponomarenko, Oleg Ieremeiev, Vladimir Lukin, Karen Egiazarian, Lina Jin, Jaakko Astola, Benoit Vozel, Kacem Chehdi, Marco Carli, Federica Battisti, et al. Color image database TID2013: Peculiarities and preliminary results. In *European Workshop on Visual Information Processing (EUVIP)*, pages 106–111, 2013. 40, 44

[101] Nikolay Ponomarenko, Lina Jin, Oleg Ieremeiev, Vladimir Lukin, Karen Egiazarian, Jaakko Astola, Benoit Vozel, Kacem Chehdi, Marco Carli, Federica Battisti, et al. Image database TID2013: Peculiarities, results and perspectives. *Signal Processing: Image Communication*, 30:57–77, 2015. 40, 44

[102] Nikolay Ponomarenko, Vladimir Lukin, Alexander Zelensky, Karen Egiazarian, Marco Carli, and Federica Battisti. TID2008: A database for evaluation of full-reference visual quality assessment metrics. *Advances of Modern Radioelectronics*, 10(4):30–45, 2009. 40, 44

[103] Abhijith Punnappurath and Michael S Brown. Learning raw image reconstruction-aware deep image compressors. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019. 39

[104] Stanislav Pyatykh, Jürgen Hesser, and Lei Zheng. Image noise level estimation by principal component analysis. *IEEE transactions on Image Processing (TIP)*, 22(2):687–699, 2013. 35

[105] Tal Remez, Or Litany, Raja Giryes, and Alex M Bronstein. Deep class-aware image denoising. In *International Conference on Sampling Theory and Applications (SampTA)*, pages 138–142, 2017. 112

[106] Tal Remez, Or Litany, Raja Giryes, and Alex M Bronstein. Class-aware fully convolutional Gaussian and Poisson denoising. *IEEE Transactions on Image Processing (TIP)*, 27(11):5707–5722, 2018. 112

[107] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1530–1538, 2015. 69, 73

[108] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241. Springer, 2015. 99, 102

[109] Stefan Roth and Michael J Black. Fields of experts. *International Journal on Computer Vision (IJCV)*, 82(2):205–229, 2009. 62, 63

[110] Mark Sheinin, Yoav Y Schechner, and Kiriakos N Kutulakos. Computational imaging on the electric grid. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 46

[111] Weng-tai Su, Gene Cheung, Richard Wildes, and Chia-Wen Lin. Graph neural net using analytical graph filters and topology optimization for image denoising. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8464–8468, 2020. 34

[112] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010. 23

[113] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4539–4547, 2017. 16, 33, 39, 93

[114] Hossein Talebi and Peyman Milanfar. Global image denoising. *IEEE Transactions on Image Processing (TIP)*, 23(2):755–768, 2014. 32, 39, 62, 63

[115] Radu Timofte, Shuhang Gu, Jiqing Wu, Luc Van Gool, Lei Zhang, Ming-Hsuan Yang, Muhammad Haris, et al. NTIRE 2018 challenge on single image super-resolution: Methods and results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018. 105

[116] Radu Timofte, Rasmus Rothe, and Luc Van Gool. Seven ways to improve example-based single image super resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1865–1873, 2016. 104

[117] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 839–846, 1998. 28

[118] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing (TIP)*, 13(4):600–612, 2004. 64, 85, 98

[119] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *Asilomar Conference on Signals, Systems & Computers*, volume 2, pages 1398–1402. IEEE, 2003. 102

[120] Andrew R Webb. *Statistical Pattern Recognition.* John Wiley & Sons, 2003. 85

[121] Jun Xu, Hui Li, Zhetong Liang, David Zhang, and Lei Zhang. Real-world noisy image denoising: A new benchmark. *arXiv preprint arXiv:1804.02603*, 2018. 40

[122] Jun Xu, Lei Zhang, and David Zhang. External prior guided internal prior learning for real-world noisy image denoising. *IEEE Transactions on Image Processing (TIP)*, 27(6):2996–3010, 2018. 40

[123] Songhyun Yu, Bumjun Park, and Jechang Jeong. Deep iterative down-up CNN for image denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019. 102

[124] Jiachao Zhang and Keigo Hirakawa. Improved denoising via Poisson mixture modeling of image sensor noise. *IEEE Transactions on Image Processing (TIP)*, 26(4):1565–1578, 2017. 20, 70, 73

[125] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing (TIP)*, 26(7):3142–3155, 2017. x, xiv, xvi, 16, 32, 39, 62, 63, 64, 66, 67, 68, 85, 87, 91, 93

[126] Kai Zhang, Wangmeng Zuo, and Lei Zhang. FFDNet: Toward a fast and flexible solution for CNN-based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018. 93

[127] Lei Zhang, Weisheng Dong, David Zhang, and Guangming Shi. Two-stage image denoising by principal component analysis with local pixel grouping. *Pattern Recognition*, 43(4):1531–1549, 2010. 62, 63

[128] Wenpian Paul Zhang and Xingyuan Tong. Noise modeling and analysis of SAR ADCs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(12):2922–2930, 2015. 12

[129] Fengyuan Zhu, Guangyong Chen, and Pheng-Ann Heng. From noise modeling to blind image denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 42

[130] CW Zitnick, Nebojsa Jojic, and Sing Bing Kang. Consistent segmentation for optical flow estimation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1308–1315, 2005. xii, 36

[131] Daniel Zoran and Yair Weiss. From learning models of natural image patches to whole image restoration. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2011. 62, 63

Appendix A

List of Publications

1. Abdelrahman Abdelhamed, Stephen Lin, and Michael S. Brown. A High-Quality Denoising Dataset for Smartphone Cameras. In *Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
   Paper | Website & Dataset | Code 1 | Code 2

2. Abdelrahman Abdelhamed, Marcus A. Brubaker, and Michael S. Brown. Noise Flow: Noise Modeling with Conditional Normalizing Flows. In *Proceedings of The IEEE International Conference on Computer Vision (ICCV)*, 2019.
   Paper | Code & Models

3. Abdelrahman Abdelhamed, Radu Timofte, Michael S. Brown, Songhyun Yu, Bumjun Park, Jechang Jeong, Seung-Won Jung, Dong-Wook Kim, Jae-Ryun Chung, Jiaming Liu, YuzhiWang, Chi-Hao Wu, Qin Xu, Yuqian Zhou, Chuan Wang, Shaofan Cai, Yifan Ding, Haoqiang Fan, Jue Wang, Kai Zhang, Wangmeng Zuo, Magauiya Zhussip, Dong Won Park, Shakarim Soltanayev, Se Young Chun, Zhiwei Xiong, Chang Chen, Muhammad Haris, Kazutoshi Akita, Tomoki Yoshida, Greg Shakhnarovich, Norimichi Ukita, Syed Waqas Zamir, Aditya Arora, Salman Khan, Fahad Shahbaz Khan, Ling Shao, Sung-Jea Ko, Dong-Pan Lim, Seung-Wook Kim, Seo-Won Ji, Sang-Won Lee, Wenyi Tang, Yuchen Fan, Yuqian Zhou, Ding Liu, Thomas S. Huang, Deyu Meng, Lei Zhang, Hongwei Yong, Yiyun Zhao, Pengliang Tang, Yue Lu, Raimondo Schettini, Simone Bianco, Simone Zini, Chi Li, Yang Wang, and Zhiguo Cao. NTIRE 2019 Challenge on Real Image Denoising: Methods and Results. In *Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.
   Paper | Website

4. Abdelrahman Abdelhamed, Mahmoud Afifi, Radu Timofte, Michael S. Brown, Yue Cao, Zhilu Zhang, Wangmeng Zuo, Xiaoling Zhang, Jiye Liu, Wendong Chen, Changyuan Wen, Meng Liu, Shuailin Lv, Yunchao Zhang, Zhihong Pan, Baopu Li, Teng Xi, Yanwen Fan, Xiyu Yu, Gang Zhang, Jingtuo Liu, Junyu Han, Errui Ding, Songhyun Yu, Bumjun Park, Jechang Jeong, Shuai Liu, Ziyao Zong, Nan Nan, Chenghua Li, Zengli Yang, Long Bao, Shuangquan Wang, Dongwoon Bai, Jungwon Lee, Youngjung Kim, Kyeongha Rho, Changyeop Shin, Sungho Kim, Pengliang Tang, Yiyun Zhao, Yuqian Zhou, Yuchen Fan, Thomas Huang, Zhihao Li, Nisarg A. Shah, Wei Liu, Qiong Yan, Yuzhi Zhao, Marcin Możejko, Tomasz Latkowski, Lukasz Treszczotko, Michał Szafraniuk, Krzysztof Trojanowski, Yanhong Wu, Pablo Navarrete Michelini, Fengshuo Hu, Yunhua Lu, Sujin Kim, Wonjin Kim, Jaayeon Lee, Jang-Hwan Choi, Magauiya Zhussip, Azamat Khassenov, Jong Hyun Kim, Hwechul Cho, Priya Kansal, Sabari Nathan, Zhangyu Ye, Xiwen Lu, Yaqi Wu, Jiangxin Yang, Yanlong Cao, Siliang Tang, Yanpeng Cao, Matteo Maggioni, Ioannis Marras, Thomas Tanay, Gregory Slabaugh, Youliang Yan, Myungjoo Kang, Han-Soo Choi, Kyungmin Song, Shusong Xu, Xiaomu Lu, Tingniao Wang, Chunxia Lei, Bin Liu, Rajat Gupta, and Vineet Kumar. NTIRE 2020 Challenge on Real Image Denoising: Dataset, Methods and Results. In *Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020.
   Paper | Website