LEVERAGING DUAL-PIXEL SENSORS FOR CAMERA DEPTH OF FIELD MANIPULATION

ABDULLAH ABUOLAIM

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 THE DEPARTMENT OF
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

YORK UNIVERSITY
TORONTO, ONTARIO
October 2021

ABSTRACT

Capturing a photo with clear scene details is important in photography and for computer vision applications. The range of distance in the real world that makes the scene's objects appear with clear details is known to be the camera's depth of field (DoF). The DoF is controlled by either adjusting lens distance to sensor (i.e., focus distance), aperture size, and/or focal length of the cameras. At capture time, especially for video recording, DoF adjustment is often restricted to lens movements as adjusting other parameters introduces artifacts that can be visible in the recorded video. Nevertheless, the desired DoF is not always achievable at capture time due to many reasons like the physical constraints of the camera optics. This leads to another direction of adjusting DoF after effect as a post-processing step. Although pre- or post-capture DoF manipulation is essential, there are few datasets and simulation platforms that enable investigating DoF at capture time. Another limitation is the lack of real datasets for DoF extension (i.e., defocus deblurring), where the prior work relies on synthesizing defocus blur and ignores the physical formation of defocus blur in real cameras (e.g., lens aberration and radial distortion). To address this research gap, this thesis revisits DoF manipulation from two point of views: (1) adjusting DoF at capture time, a.k.a. camera autofocus (AF), within the context of dynamic scenes (i.e., video AF); (2) computationally manipulating the DoF as a post-capturing process. To this aim, we leverage a new imaging sensor technology known as the dual-pixel (DP) sensor. DP sensors are used to optimize camera AF and can provide good cues to estimate the amount of defocus blur present at each pixel location. In particular, this thesis provides the first 4D temporal focal stack dataset along with AF platform to examine video AF. It also presents insights about user preference that lead to propose two novel video AF algorithms. As for post-capture DoF manipulation, we examine the problem of reducing defocus blur (i.e., extending DoF) by introducing a new camera aperture adjustment to collect the first dataset that has images with real defocus blur and their corresponding

all-in-focus ground truth. We also propose the first end-to-end learning-based defocus deblurring method. We extend image defocus deblurring to a new domain application (i.e., video defocus deblurring) by designing a data synthesis framework to generate realistic DP video data through modeling physical camera constraints, such as lens aberration and redial distortion. Finally, we build on top of a data synthesis framework to synthesize shallow DoF with other aesthetic effects, such as multi-view synthesis and image motion.

To my parents

ACKNOWLEDGMENT

الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ وَالصَّلاَةُ وَالسَّلاَمُ عَلَى نَبِيِّنَا مُحَمَّد وَعَلَى آلِهِ وَصَحْبِهِ أَجْمَعِين

In the Name of Allah, the Most Gracious, the Most Merciful. Praise belongs to Allah alone; prayers and peace be upon the last Prophet Mohamed His servant and messenger.

First and foremost, I would like to express my greatest gratitude to my father, Ahmad Abuolaim; Allah rest his soul. I would not be the person I am today, without his love and constant encouragement; I love you and miss you every day. To my beloved mother, Ni'mat Al-Bzour, who has always been my inspiration and strength to continue. I would also like to express my appreciation to my brother, Ali, and my sisters, Nour, Rasha, Rama, and Tasneem; to my best friend, Ahmad Dughmi; to my uncle, Hasan Al-Bzour, for their endless love, support, and encouragement.

A special thank you to my supervisor, Michael S. Brown, for his continuous support and guidance; he supported my passion and growth, not only in my work but as a person too. He has always been a responsible advisor and I will always be grateful for the opportunity to learn beside him. I also thank my thesis advisory committee, Marcus A. Brubaker and Richard P. Wildes, for their guidance throughout my Ph.D. journey.

My success at York University would not be possible without the friends I have met over the past years. I want to take this opportunity to show my sincere appreciation to my friends and colleagues; Enas Altarawneh, Justin Bortolin, Marco Tosato, Mahmoud Afifi, Abdelrahman Abdelhamed, Abhijith Punnappurath, Abbas Masoumzadeh, Calden Wloka, Hakki Can Karaimer, Markus Solbach, Saim Mehmood, Wenxiao Fu, Niloy Eric Costa, Hoang Le, and Sadeq Bani Melhem for always being there supporting, encouraging, and helping me when I most needed it. I would also like to thank my research mentors, Manos Papagelis, Mauricio Delbracio, Damien Kelly, and Peyman Milanfar, for their effort and constructive discussions.

I am very proud of my accomplishments, and I would not change any event that took place in my journey. With each success there were countless failures I overcame, each one made me into a stronger researcher, and ultimately a Doctor!

Contents

List of Figures

List of Tables

List of Acronyms

AF	Autofocus
API	Application Programming Interface
BLSTM	Bidirectional Long Short Term Memory
CBAM	Convolutional Block Attention Aodule
CDAF	Contrast Detection Autofocus
CG	Computer Generated
CNN	Convolutional Neural Networks
CoC	Circle of Confusion
conv	Convolutional
convLSTM	Convolutional Long Short Term Memory
CPU	Central Processing Unit
CST	Color Space Transformation
dB	Decibel
DC	Direct Current
DCT	Discrete Cosine Transform
DFD	Depth From Defocus
DMENet	Defocus Map Estimation Network
DNN	Deep Neural Network
DNG	Digital Negative
DoF	Depth of Field
DP	Dual-Pixel
DPDNet	Dual-Pixel Deblurring Network
DPRR	Dual-Pixel Reflection Removal
DSLR	Digital Single Lens Reflex

EBDB	Edge-Based Defocus Blur
FB	Face in Background
FF	Face in Foreground
FoV	Field of View
FR	Face Region
GB	Global
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
GT	Ground Truth
GUI	Graphical User Interface
ISP	Image Signal Processor
JPEG	Joint Photographic Experts Group
JNB	Just Noticeable Blur
LED	Light-emitting Diode
ReLU	Leaky Rectified Linear Unit
LPIPS	Learned Perceptual Image Patch Similarity
LReLU	Leaky Rectified Linear Unit
LSTM	Long Short Term Memory
MA	Moving Average
MAE	Mean Absolute Error
MB	Megabyte
MDP	Multi-Task Dual-Pixel
MSE	Mean Squared Error
MSSSIM	Multi-Scale Structural Similarity Index Measure
N/A	Not Applicable
NF	No Face
OoF	Out-of-Focus
PDAF	Phase Difference Autofocus
PCG	Preconditioned Conjugate Gradient
PR	Precision-Recall
PSF	Point Spread Function
PSNR	Peak Signal-to-Noise Ratio

RCN	Recurrent Convolutional Network
RDPD	Recurrent Dual-Pixel Deblurring
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
RoI	Region of Interest
RSPD	Recurrent Single-Pixel Deblurring
SAM	Supervised Attention Module
SDK	Software Development Kit
SG	Savitzky-Golay
SRNet	Scale-Recurrent Network
SSD	Sum of Squared Differences
SSIM	Structural Similarity Index Measure
ToFAF	Time-of-Flight Autofocus
ULSTM	Unidirectional Long Short Term Memory
UPI	Unprocessing Technique
w.r.t	with respect to
WMA	Weighted Moving Average
9 FP	9 Focus Points
51 FP	51 Focus Points

List of Publications

1. **Abdullah Abuolaim**, Mahmoud Afifi, and Michael S. Brown. "Multi-View Motion Synthesis via Applying Rotated Dual-Pixel Blur Kernels". In the *Winter Conference on Applications of Computer Vision Workshops (WACVW)*, 2022.

2. **Abdullah Abuolaim**, Mahmoud Afifi, and Michael S. Brown. "Improving Single-Image Defocus Deblurring: How Dual-Pixel Images Help Through Multi-Task Learning". In the *Winter Conference on Applications of Computer Vision (WACV)*, 2022.

3. **Abdullah Abuolaim**, Mauricio Delbracio, Damien Kelly, Michael S. Brown, and Peyman Milanfar. "Learning to Reduce Defocus Blur by Realistically Modeling Dual-Pixel Data". In the *International Conference on Computer Vision (ICCV)*, 2021.

4. **Abdullah Abuolaim**, Radu Timofte, Michael S. Brown, et al. "NTIRE 2021 Challenge for Defocus Deblurring Using Dual-Pixel Images: Methods and Results". In the *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021.

5. Mahmoud Afifi, Abdelrahman Abdelhamed, **Abdullah Abuolaim**, Abhijith Punnappurath, and Michael S. Brown. "CIE XYZ Net: Unprocessing Images for Low-Level Computer Vision Tasks". In the *IEEE Transactions on Pattern Analysis andMachine Intelligence (TPAMI)*, 2021.

6. **Abdullah Abuolaim** and Michael S. Brown. "Defocus Deblurring Using Dual-Pixel Data". In the *European Conference on Computer Vision (ECCV)*, 2020.

7. Abhijith Punnappurath, **Abdullah Abuolaim**, Mahmoud Afifi, and Michael S. Brown. "Modeling Defocus-Disparity in Dual-Pixel Sensors". In the *IEEE International Conference on Computational Photography (ICCP*), 2020.

8. **Abdullah Abuolaim** and Michael S. Brown. "Online Lens Motion Smoothing for Video Autofocus". In the *Winter Conference on Applications of Computer Vision (WACV)*, 2020.

9. **Abdullah Abuolaim**, Abhijith Punnappurath, and Michael S. Brown. "Revisiting Autofocus for Smartphone Cameras". In the *European Conference on Computer Vision (ECCV)*, 2018.

List of Patent Applications

1. **Abdullah Abuolaim**, Mahmoud Afifi, Michael S. Brown. "System and Method of Dual-pixel Image Synthesis and Image Background Manipulation". USPR Patent Application, 2021.

2. Michael S. Brown, Mahmoud Afifi, Abdelrahman Abdelhamed, Hakki Can Karaimer, **Abdullah Abuolaim**, and Abhijith Punnappurath. "System and Method of Processing of a Captured Image to Facilitate Post-processing Modification". Worldwide Patent Application, 2020.

Part I

Introduction and Preliminaries

Chapter 1

Introduction

(a) Out of focus (b) In focus

Figure 1.1: Capturing photos with proper camera focus is important and produces high-quality photos. The photo on the left is out of focus and the one on the right is taken with proper camera focus (i.e., in focus).

Capturing a photo with proper *camera focus* is an important aesthetic quality of photographs (see Figure 1.1). *Camera focus* is a physical process that occurs at the hardware/optics level during capture time. This process is not straightforward as it incorporates many factors to decide the range of distance in the real world that makes the scene's objects appear in sharp focus. This range of distance is known to be the camera's depth of field (DoF). The DoF can be controlled by either adjusting the camera aperture, focal length, and/or lens distance to sensor (i.e., focus distance). Changing the aperture size or focal length leads to expanding or shrinking the DoF, whereas changing lens position

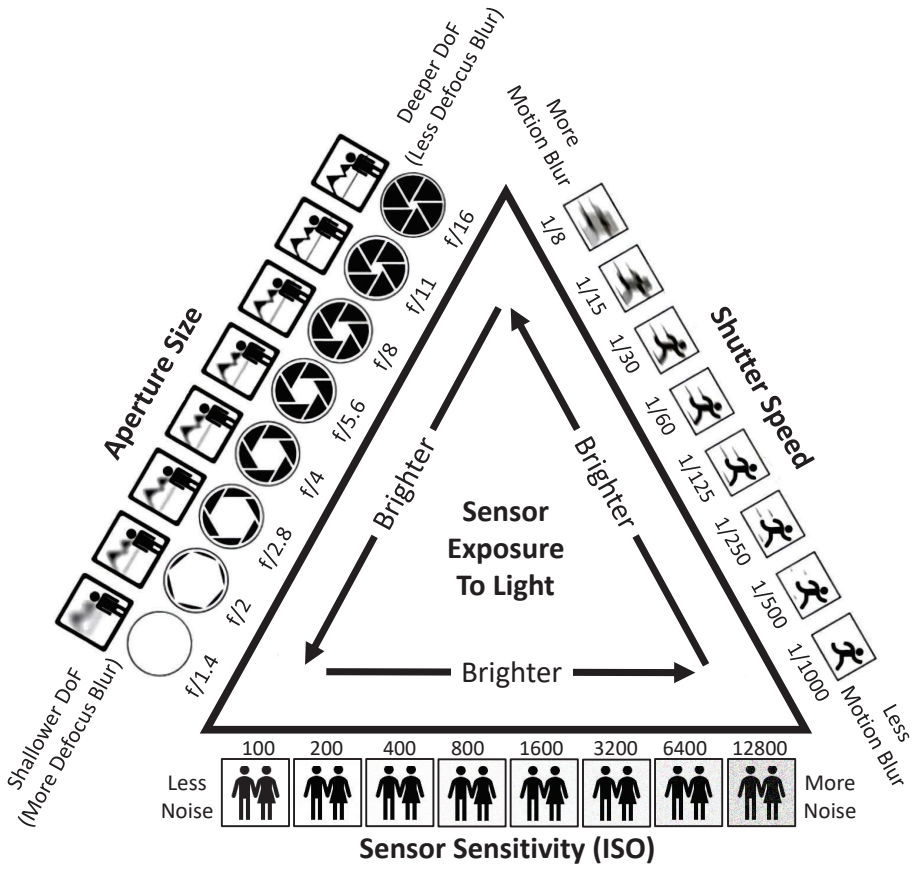


Figure 1.2: Exposure triangle. Imaging sensor exposure to light can be controlled through three camera parameters: shutter speed, aperture size, and sensor's sensitivity to light (i.e., ISO). This triangle illustrates the relationship between these parameters and the effect on photo exposure. It also shows the possible image degradation like motion blur, defocus blur, and noise. Note: we acknowledge that this figure was adapted from Helen Hooker's blog in PhotoBlog (`https://www.photoblog.com/learn/exposure-triangle-guide/`).

results in shifting the DoF.

Adjusting camera DoF is usually constrained with the scene conditions and camera settings to capture properly exposed photos (i.e., sufficient brightness). Photo exposure to light can be controlled by adjusting any of three camera parameters: shutter speed, aperture size, and sensor's sensitivity to light (i.e, ISO). The shutter speed controls the duration of light falling on the sensor, while the aperture controls the amount of light passing through the optics. The ISO controls the signal gain of the camera's sensor. The

reciprocity between these three parameters allows the same exposure to occur by fixing one parameter and adjusting the rest. For example, when a camera is set in *aperture-priority* mode with fixed ISO, the aperture remains fixed while the shutter speed is adjusted to control how long light is allowed to pass through the lens. The drawback is that a slow shutter speed can result in motion blur if the camera and/or an object in the scene moves while the shutter is open. Conversely, in *shutter-priority* mode with fixed ISO, the shutter speed remains fixed while the aperture adjusts its size. The drawback of a variable aperture is that a wide aperture results in a shallow DoF, causing defocus blur to occur in scene regions outside the DoF. Additionally, increasing ISO results in larger photo exposure (i.e., brightness), however, higher ISO leads to more noise. The relation between these three camera parameters and the effect on photo exposure is referred to as *exposure triangle* (illustrated in Figure 1.2).

The DoF adjustment is also related to the camera type or imaging system. This thesis focuses on the most common types of consumer cameras: (1) the digital single lens reflex (DSLR) camera and (2) a smartphone camera. For both cameras, we utilize the new imaging sensor technology, i.e., dual-pixel (DP) sensor. DP sensors are used for better control of camera focus by capturing two sub-aperture views of the scene in a single image shot. The two sub-aperture views are used to calculate the appropriate lens position to focus on a particular scene region and are discarded afterwards. In this thesis, we utilize these discarded sub-aperture views for DoF manipulation.

A DSLR, with a controllable aperture and optics, can offer flexible control of DoF at capture time. On the other hand, smartphone cameras are limited to lens movements only for DoF adjustments and provide less options to control DoF, since they are made to be small and portable in a single hand. They are also general-purpose devices and the physical space assigned to the camera is very small, which makes manufacturing sophisticated on-board imaging systems, similar to DSLRs, impossible.

While the DoF, of a DSLR or smartphone camera, can be controlled either manually or automatically, there are cases where the desired DoF cannot be achieved at capture time. These cases occur due to many reasons like physical limitations of the camera's optical system (e.g., smartphone cameras) or capturing constraints (e.g., capturing a fast-moving object in low-light conditions that requires large camera exposure with fast shutter speed). It is even more challenging when it comes to controlling DoF for dynamic scenes while recording a video, where the camera and scene's objects are moving. Not only that,

but also the only mechanism to adjust camera focus during video record are the lens movements, since changing aperture size or focal length introduces artifacts. For example, adjusting aperture while recording a video produces a noticeable change in exposure that introduces an unpleasant effect.

There are also many research directions related to DoF manipulation that have not been explored, and many research questions are left open. For example, which part of the scene is preferred by users to be within camera's DoF? When and how to move the camera lens in the context of dynamic scenes? What are the benefits of having scene details in sharp focus and what is the effect on other computer vision tasks? All these questions and others have inspired us to start the effort of this thesis. To this end, we address DoF manipulation based on DP sensor from two point of views: (1) controlling DoF at capture time (Part II), within the context of dynamic scenes (i.e., video); (2) computationally adjusting the DoF as a post-capture processing (Part III).

Pre-capture DoF manipulation The main application of pre-capture DoF manipulation is camera autofocus (AF). Camera AF for single image capture is a straightforward process, where the user can select the region of interest (RoI) and adjust the focus then capture the image. The single image AF is also a well-established research area and many algorithms have been proposed [6–12]. However, the research area of video AF is not well studied (e.g., [13]). Not only that, but also the AF algorithms designed by camera manufacturers are not available for the public. Additionally, we found that different cameras have different video AF behavior when recording the same dynamic scene [14]. Therefore, in this thesis, we revisit the problem of camera AF and address video AF issues within the context of dynamic scenes. To start the effort, we find it necessary to capture a new temporal focal stack dataset and develop an AF platform with its associated application programming interface (API) (details in Chapter 3). Then, based on the initial findings of our AF user study, we propose novel video AF algorithms that aim to reduce unnecessary lens motion without affecting scene sharpness (Chapter 4). To our knowledge, our video AF dataset, platform, API, and algorithms are the first in the research community and can facilitate further exploration of the video AF area.

Post-capture DoF manipulation Despite having a reliable AF system, the desired DoF is not always achievable at capture time [3]. Therefore, there is an obvious need

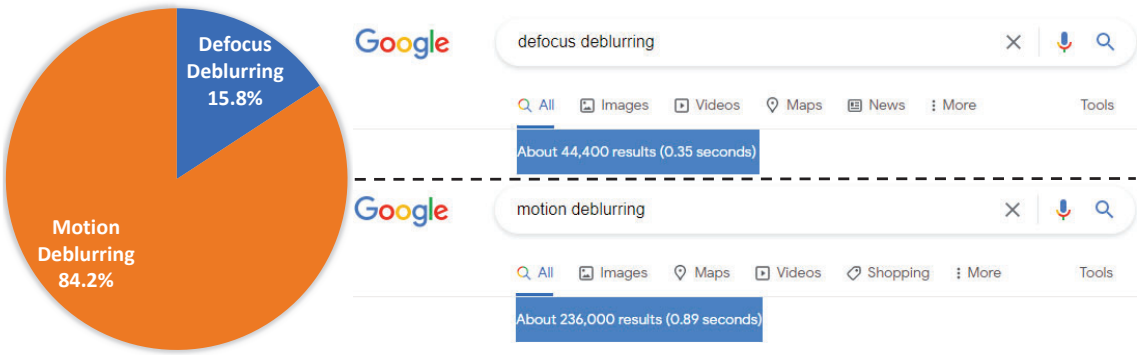


Figure 1.3: The percentages of Google search results for defocus deblurring vs. motion deblurring. The research problem of defocus deblurring receives much less attention compared to motion deblurring.

for a post-processing DoF adjustment. This thesis addresses adjusting DoF after effect in two directions, namely, DoF extension (Chapter 5–7) and DoF synthesis (Chapter 8). Although DoF extension (or *defocus deblurring*) is as important as *motion deblurring* (e.g., [2, 15–22]), we find much less research on it as the main focus is on topics related to reducing motion blur. Figure 1.3 shows the percentages of the research results when Google searching *motion deblurring* vs. *defocus deblurring*. To encourage research on defocus deblurring, we begin the effort by introducing a new procedure to collect the first real DP defocus blur dataset that comes with ground truth all-in-focus reference images. In addition to the new dataset, we propose an end-to-end learning-based framework to reduce defocus blur in Chapter 5. We also demonstrate the utility of DP in reducing defocus blur and how other computer vision tasks can benefit from the deblurred images that have sharper details.

Chapter 6 addresses the limitation of collecting real DP data, with its ground truth in the context of image sequences (i.e., video), by thoroughly modeling the DP image acquisition process that allows realistic generation of synthetic DP data. With this DP data synthesis, we can extend to a new application domain i.e., video defocus deblurring. We are the first who address reducing defocus blur in video data, where we propose a learning-based spatiotemporal framework in Chapter 6. In Chapter 7, we examine the case of reducing defocus blur when the DP data is not available at inference time. Specifically, we propose a multi-task learning framework that takes a single input image with defocus blur to produce a sharper version of the image with its high-quality DP views. Our

multi-task framework is scalable in that it can generate more than two views (e.g., four, eight, etc.). With multiple generated views, we introduce a novel aesthetic image motion effect. Lastly, Chapter 8 introduces a way for synthesizing shallow DoF based on our DP modeling. Note that, in this thesis, we use *defocus deblurring* and DoF extension interchangeably as they share the same definition (i.e., extending DoF means reducing defocus blur).

Thesis Outline The remainder of this thesis is organized as follows, Chapter 2 reviews preliminaries and related work, including AF for cameras (Section 2.1), time-series data smoothing methods (Section 2.2), defocus deblurring methods (Section 2.3), and existing datasets (Section 2.4). Then, Part II discusses our pre-capture DoF manipulation work, including the AF dataset and platform in Chapter 3 and the online lens motion smoothing for video AF in Chapter 4. Afterwards, our contributions related to post-capture DoF manipulation are introduced in Part III (i.e., Chapter 5–8). In particular, Chapter 5 presents a novel approach for data capture along with a new dataset and image defocus deblurring algorithm. This is followed by an extended image sequence deblurring method (Chapter 6) that handles video data through a realistic modeling of defocus blur. In Chapter 7, a multi-task approach is introduced to address reducing defocus blur when the DP data is not available at inference time. Then, a framework to synthesize shallow DoF is presented in Chapter 8. At the end, an overall summary of this thesis and possible research directions are provided in Chapter 9.

Chapter 2

Preliminaries and Prior Work

This chapter discusses preliminaries and prior work related to camera AF, data smoothing and filtering methods, defocus deblurring methods, and research datasets targeting image focus/defocus.

2.1 AF for Cameras

AF systems in cameras can be categorized into two classes: *active* and *passive*. *Active* AF systems send an active signal towards the scene's objects such as laser, light-emitting diode (LED), or sonar. The reflection of these signals coming back from objects in the scene is received by the camera's main sensor or another auxiliary sensor and analyzed to determine the objects' distance from the camera and correspondingly adjust the focus. In *passive* AF systems, nothing is sent towards the scene. Instead, the scene is captured under its own ambient illumination and the image is analyzed in order to adjust the focus of a certain RoI. For dark scenes, some *passive* AF systems provide additional illumination during focusing, so-called *AF assist light* or *camera flash*. In this section, we provide a review of existing *active* and *passive* AF technologies including: the time-of-flight autofocus (ToFAF), phase difference autofocus (PDAF), and contrast detection autofocus (CDAF).

2.1.1 Time-of-Flight Autofocus

The ToFAF technology is one of the most common *active* AF technology used for cameras. ToFAF incorporates an auxiliary light sensor and uses time-of-flight techniques to determine the distance between the camera and the scene's object of interest. Specifically,

the distance is determined by measuring the round trip time of an artificial light signal provided by a laser or an LED [6, 7]. Once the distance from the object of interest is calculated, the ToFAF system can predict the optimal in-focus lens position that maximizes the RoI's sharpness and brings it into sharp focus by moving the lens. The camera's lenses are moved in discrete steps using a stepper motor.

The ToFAF systems aim to calculate distance from the object of interest, which makes them able to provide an accurate focusing for both low light scenes and homogeneous regions. However, the light emitting range is limited for the ToFAF systems. They are also sensitive to noise from other light sources, thus they tend to fail in calculating distance for outdoor scenes.

2.1.2 Phase Difference Autofocus

The PDAF system is a widely employed *passive* AF technology found in most of DSLR cameras and smartphone cameras. The PDAF technology operates at a hardware/optics level and aims to adjust the lens position such that the phase between two light rays coming from a scene point is matched. The PDAF hardware module can be designed in three ways: (1) half sub-mirror with line sensor as used in older DSLR cameras [8, 9], (2) on-sensor scattered pixels layout [10, 11], and (3) on-sensor dual-pixel layout used in most of modern DSLR and smartphone cameras [14, 23].

Half sub-mirror with line sensor A half sub-mirror with line sensor PDAF is the most common type used in older DSLR cameras. This approach has two small sensors, each one has a micro lens above it. These sensors are typically one-dimensional, i.e. a row of pixels (usually called line sensors) arranged in pairs. Each pair refers to one AF point, which looks at one part of the camera's field of view (FoV).

Figure 2.1 illustrates how this technology works. In this figure, we show a pair of line sensors of one AF point. The line sensors receive the light rays coming from the same point of the camera's FoV. Based on the 1D signal, the phase difference (i.e., difference in signal) between the left and right line sensors are calculated as shown in the red and green curves below. Since these two curves are separated by some distance (even if only a few millimeters), they will see slightly different views. In this figure, we also show the three possible focusing cases i.e., front-focus (Figure 2.1-top), in-focus (Figure 2.1-middle), rear-focus (Figure 2.1-bottom). The intersection position of the light rays–i.e.,

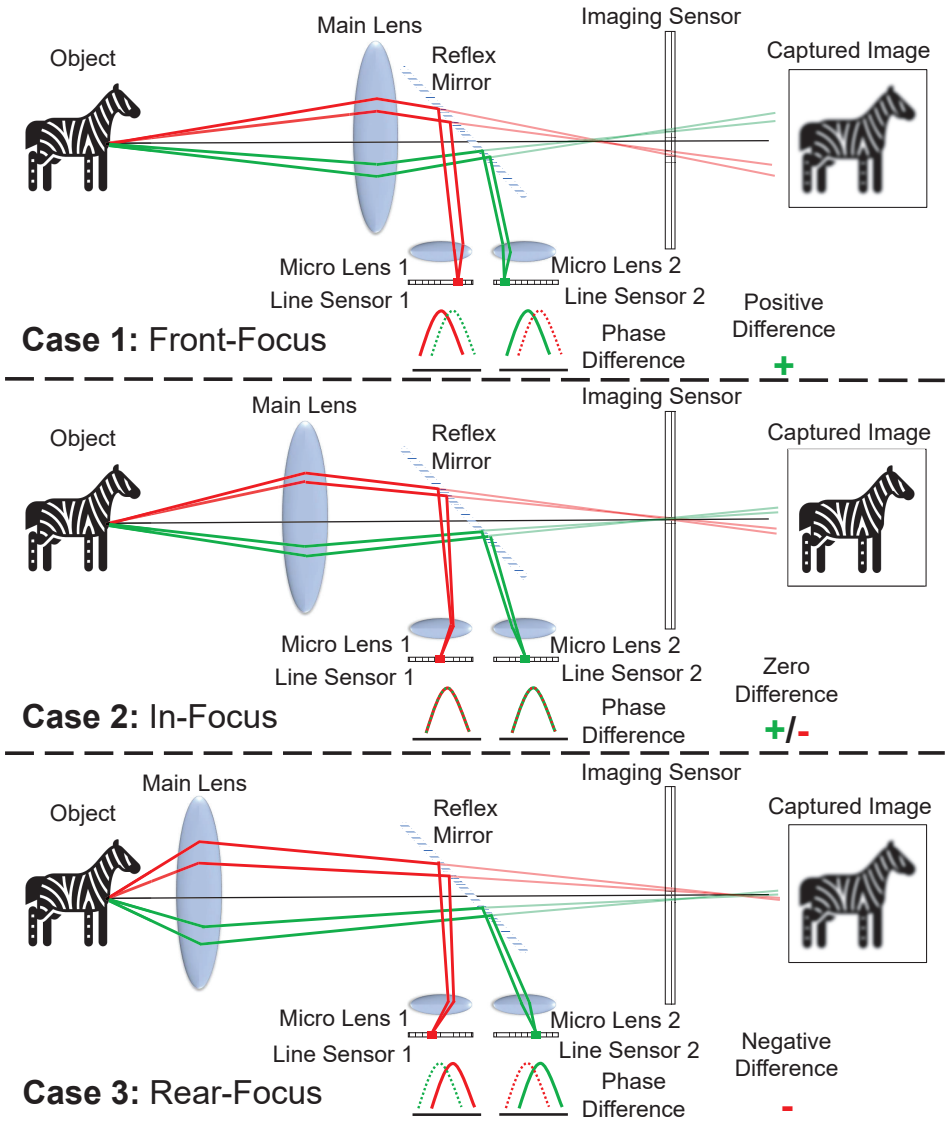


Figure 2.1: An abstract camera model illustrates how the half sub-mirror with line sensor PDAF works. Pair of line sensors are used to calculate the phase difference of different light rays coming from one part of the camera's field of view. The intersection position of these light rays represents the different focusing cases. Note that the reflex mirror contains a partially silvered section. This allows a portion of the light to pass through.

red ray coming from the upper part and green ray coming from the lower part of the main lens–on the optical axis represents the focusing cases. As shown, in the captured image on the right, the point from the scene is in-focus only when the light rays intersect exactly

at the imaging sensor; otherwise, it is out-of-focus. Note that the reflex mirror contains a partially silvered section. This allows a portion of the light to pass through.

This technology is able to approximate the optimal lens position in a single processing step at a hardware level in a fast way. However, this technology can only be used when the reflex mirror is down. Thus, in live preview mode, in which the mirror is flipped up, the camera cannot use PDAF. In this mode the camera either offers no autofocusing at all or it offers only CDAF, which we explain in Section 2.1.3.

On-sensor scattered pixels layout More modern cameras use an on-sensor layout technology that functionally works in a similar way to the half sub-mirror with line sensor PDAF. The only difference is that the on-sensor layout PDAF technology does not require any additional optical parts, reflex mirrors, and focusing line sensors, instead it utilizes the imaging sensor for phase difference calculations [10]. Therefore, the on-sensor layout PDAF technology made it possible for smartphone cameras to employ PDAF with their limited physical dimensions. The on-sensor layout PDAF has a special imaging sensor where there are some pixels that have two separate light-sensitive photodiodes and each one of those pixels belongs to a single AF point. These photodiodes convert light into an electronic signal used for phase difference calculations.

On-sensor dual-pixel layout Both the half sub-mirror with line sensor PDAF and the on-sensor layout PDAF are designed in a way that they assign a fixed and static AF points (i.e., RoIs) inside a camera's FoV. For example, most of the cameras that employ any of these technologies, assign the center region as their RoI and the users have to move the camera to keep the desired object they want in focus in the center.

In order to provide an imaging sensor that enables dynamic RoIs, Canon introduced the new dual-pixel (DP) imaging sensor [24]. Each pixel on the DP imaging sensor has two separate, light-sensitive photodiodes, which convert light into an electronic signal. Independently, each half of a pixel detects light through separate micro lenses, atop each pixel. During the process of PDAF, the two halves of each pixel–the two photodiodes–send separate signals, which are used for phase difference calculations to find the optimal lens position.

Figure 2.2 shows an abstract camera model that employs DP technology. Since each pixel on the DP imaging sensor has two separate photodiodes, the DP sensor can be

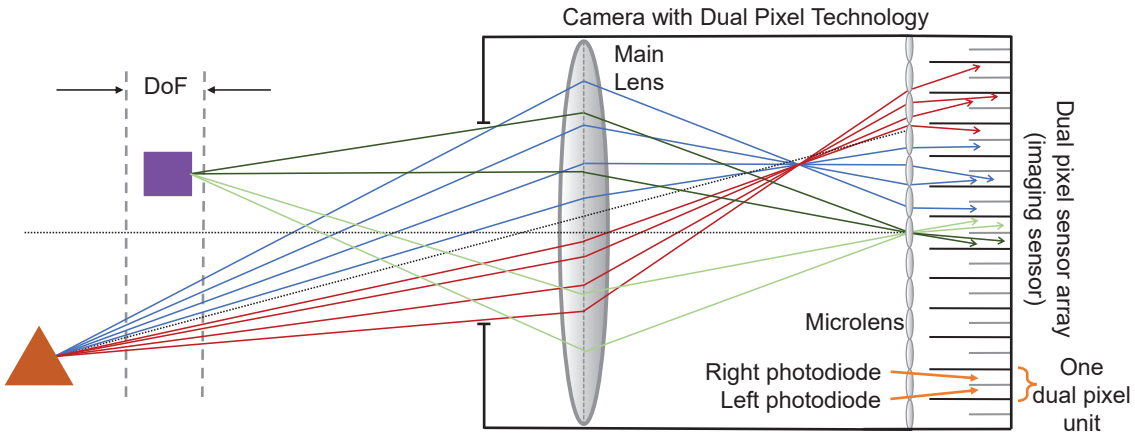


Figure 2.2: An abstract camera model that employs dual pixel technology. Each pixel on the dual pixel AF imaging sensor has two separate light-sensitive photodiodes. The purple box is within camera's DoF which makes it to appear in-focus in the captured image. The Brown triangle is outside camera's DoF and will appear out-of-focus in the captured image. Note: this figure is adapted from [5].

shown as two-sample light-field (left and right views) with an approximate one millimeter baseline. Within this context, scene points that are within the camera's DoF (in-focus) will have no difference between their positions in the left and right views. However, out-of-focus scene points will be blurred in opposite directions in the two views, resulting in very small but detectable shifts. These shifts, which we refer to as defocus-disparity cues, are correlated to the amount of out-of-focus blur incurred by the scene point with respect to the camera lens DoF.

DP sensors also perform PDAF, in which the phase difference between the left and right sub-aperture views of the primary lens is calculated to measure the blur amount. Using this phase information, the camera's lens is adjusted such that the blur is minimized. While intended for autofocus, the DP images have been found useful for other tasks, such as depth map estimation [4, 25, 26], reflection removal [5], and synthetic DoF [27].

2.1.3 Contrast Detection Autofocus

The CDAF technology is one of the most basic AF techniques used in almost all camera types including smartphone, DSLR, point-and-shoot cameras, etc. CDAF operates by applying low-level image processing algorithms (e.g., gradient magnitude analysis) to

determine the sharpness of a *single* image or RoI in an image [12]. CDAF works in three steps: (1) RoI selection, (2) RoI sharpness measure, and (3) search algorithm. For the RoI selection, cameras use a finite set of AF basic RoIs; e.g., center focus, 51 focus points, face priority by running some face detection algorithms. Once the RoI is selected, the second step is to measure the RoI sharpness in the captured image at the current lens position. Afterwards, the lens is moved back and forth in order to find the lens position that has the maximum calculated sharpness of the selected RoI.

Another AF term that is connected to image capture and lens movements is the *focal stack* term. A focal stack is a set of images captured at different lens positions or focus distances. In this thesis, we also use the term *full focal stack*, in which the images are captured at discrete focus distances from the minimum to the maximum lens position allowed.

Sharpness measure Many different sharpness measures have been proposed using image gradient [28, 29], discrete cosine transform (DCT) [30, 31], and Bayesian spectral entropy [32]. Several surveys exist that examine different sharpness measures performance under various conditions [29, 33].

There are two common image gradient-based operators used in the literature, namely, Sobel and Prewitt discrete differentiation operators [28, 29]. These operators are used to compute an approximation of the gradient of the RoI/image intensity function. Both operators adopt a 2D spatial gradient measurement for sharpness calculation by convolving the RoI/image with a small, separable, and integer-valued kernel in the horizontal and vertical directions. For each pixel in the RoI/image, these operators determine two gradient values based on the horizontal and vertical neighborhood pixels, and then the overall gradient value G can be determined as follows:

$$G = \sum_{i=1}^{M} \sum_{j=1}^{N} \sqrt{G_x(i,j)^2 + G_y(i,j)^2}, \tag{2.1}$$

where $G_x(i,j)^2$ and $G_y(i,j)^2$ are the horizontal and vertical gradient values, respectively, of the RoI/image pixel at i, j. The only difference between Sobel and Prewitt operators is that they use different kernels to convolve the input RoI/image. Since both Sobel and Prewitt operators apply separable computation, they can parallelize the process and imply fewer arithmetic computations for each RoI/image pixel. However, these operators

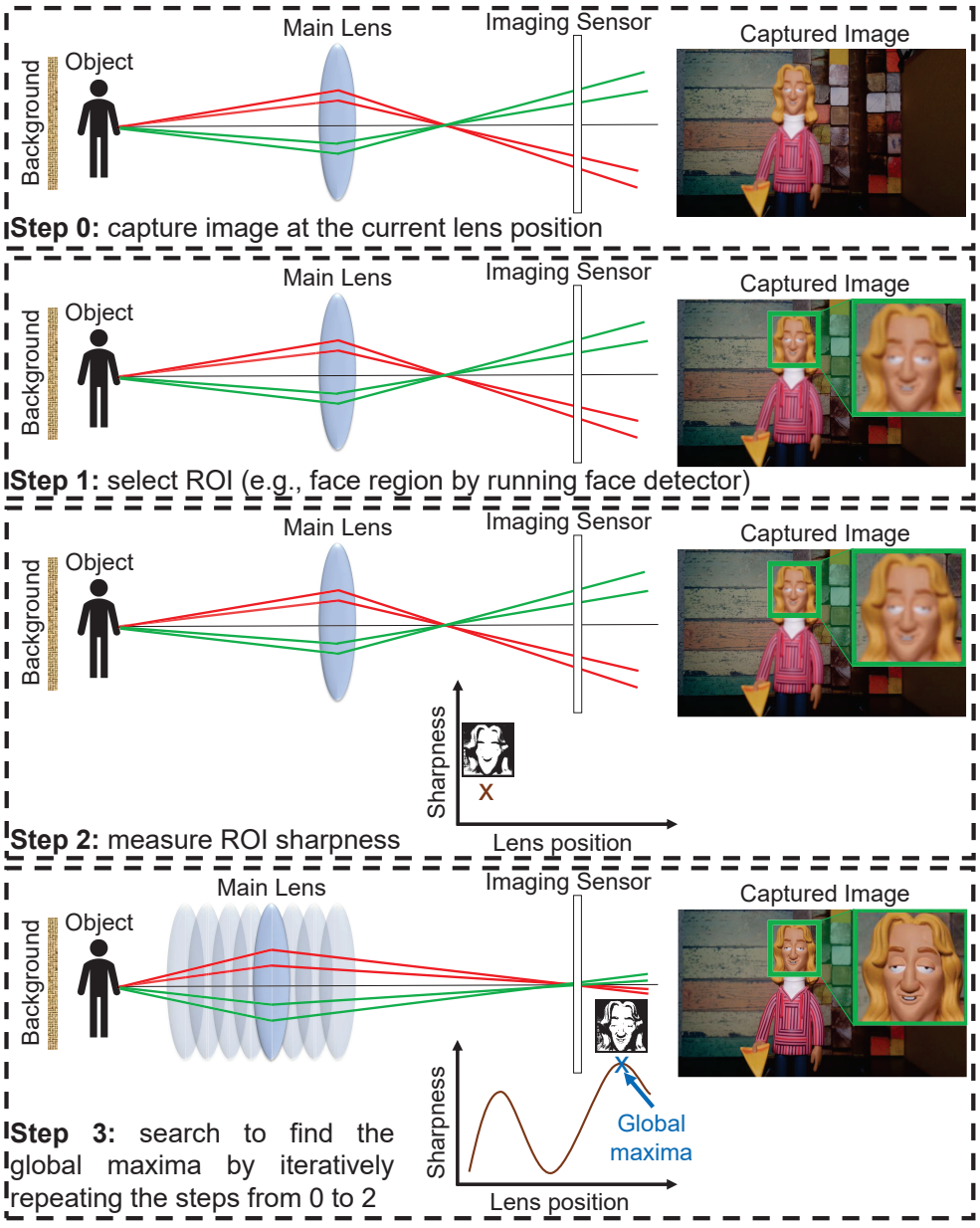


Figure 2.3: An illustrative example explains the process of the CDAF in three steps. After capturing the image at the current lens position, the first step is to select the RoI by running a computer vision algorithm (i.e., face detector). In the second step the sharpness value is calculated for the selected RoI. Afterwards, the global maxima is determined in the last step by iteratively repeating the steps from 0 to 2.

are sensitive to image noise but are still of sufficient quality to be of practical use in some applications like AF.

The DCT methods [30, 31] can be shown as one-to-one mappings between the spatial and frequency domains. They express the input RoI/image in a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies (called the DCT coefficients). In particular, these methods transform the RoI/image into the frequency domain and then use the high-frequency DCT coefficients for sharpness calculations. The DCT based methods are simple and famous for being a part of the JPEG standard compression [34]. Since most consumer cameras already incorporate JPEG standard compression, the DCT unit, thus, can be utilized for AF purposes. Compared to image gradient-based methods, the DCT methods tend to produce a uniform spectrum for the sharp RoI/image, in which they introduce multiple sharpness peaks that requires further processing to find the sharpest. To overcome this DCT defect, Kristan et al. [32] introduced a new measure of focus based on estimating the uniformity of the normalized DCT spectrum using the Bayes entropy.

The last step that CDAF performs is to search for the lens position with the sharpest RoI based on sharpness measures calculated by iteratively moving the lens [35]. Because CDAF works on a single image, the camera lens needs to be moved back and forth until the image sharpness measure is maximized [36]. Figure 2.3 shows an illustrative example of how the CDAF technique works.

2.1.4 Summary

Unlike ToFAF and PDAF, the CDAF technique is able to determine higher-level information about scene content by applying computer vision algorithms to investigate the RoI in a semantic way (e.g., face detection to detect face region). ToFAF and PDAF work at the hardware level and no analysis is applied with regard to scene content. However, CDAF requires image capture in order to apply computer vision and image processing techniques. ToFAF and PDAF do not require image capture and are able to approximate the optimal lens position in a single processing step at the hardware level in a much faster way compared to CDAF. The diagram in Figure 2.4 shows the categorization of AF technologies.

Most of the recent cameras use so-called *hybrid* AF that utilizes both PDAF and

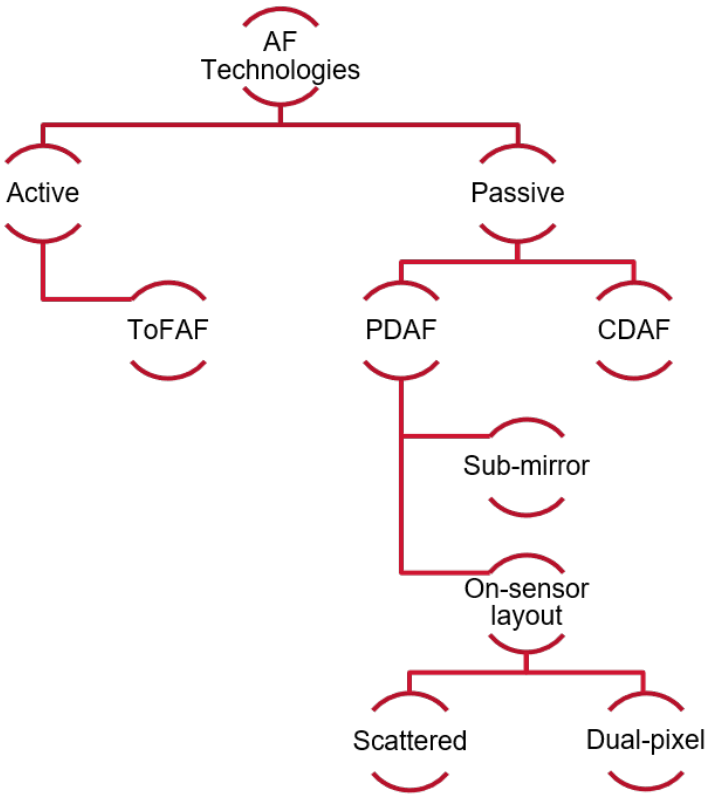


Figure 2.4: The categorization of active and passive AF technologies.

CDAF. In particular, the *hybrid* AF performs PDAF first to move the lens to a position close to the target focusing position and then performs CDAF for higher-level focusing methods based on scene content [37] (e.g., face region using face detection methods). Since CDAF applies higher-level algorithms to select the RoI, CDAF has the priority to set the RoI over the PDAF.

Prior work using both CDAF and PDAF have primarily targeted single image capture. There has been significantly less work on AF for video. Proposed methods for video AF (e.g. see [38–40]) apply the single-image approach–that targets maximum sharpness–to each frame and overlooks the fact that the lens motion can be observed in the captured video. Too much motion can result in undesired 'lens wobble'. Therefore, in Chapter 4 we propose a method to optimize both image sharpness and lens movement for video AF. We believe that this is the first work that targets this problem. The major limitation for research targeting AF for video had been the lack of access to a dataset that provides a full focal stack at each time point. In this thesis, we revisit the problem of video AF

by introducing the first 4D temporal focal stack dataset along with an AF platform that enables mimicking AF found in cameras (Chapter 3).

There are *active* AF methods that use auxiliary hardware, such as laser depth sensors; however, this thesis focuses only on *passive* AF methods that rely on data captured from the image sensor. Similar to the vast majority of recent cameras, our camera incorporates *hybrid* AF and has on-sensor DP layout PDAF where the optimal lens position that would maximize focus is known at each new input video frame. For our CDAF, we adopt the Sobel operator for measuring the RoI/image sharpness. Based on Loren's findings in [29], the Sobel operator is the most accurate among other sharpness measures.

2.2 Data Smoothing and Filtering Methods

As mentioned earlier, DoF adjustment is often restricted to lens movements when it comes to video AF. In Chapter 4, we examine reducing unnecessary lens movements without affecting overall image sharpness by smoothing lens motion over time. Therefore, in this section, we give an overview of the methods used for prepossessing time-series data including data fluctuation smoothing and data noise filtering. Several surveys exist that examine smoothing and filtering methods, readers are referred to [41–43].

2.2.1 Data Smoothing

Data smoothing is a prevalent technique used in all areas of science. Data smoothing aims to reduce undesired signal fluctuation while preserving higher-order moments of the original signal (i.e., smoother signal and the heights of spikes are lessened). There are two commonly used data smoothing methods, namely, the moving average (MA) and Savitzky-Golay (SG).

The moving average (MA) The MA is the simplest digital smoothing method [44]. This method smooths the signal by taking the averages of each sub-set in successive sub-sets of adjacent data points. These successive sub-sets can be formed by sliding a time window of size l (odd number) and stride of 1. Therefore, the smoothed data point x^* is the average value of l neighborhood data points centered around x of the original signal. Despite its simplicity, the MA is optimal for certain tasks, such as reducing random fluctuation.

Savitzky-Golay (SG) The SG is an established data smoothing method [1]. The SG has properties in common with the MA, while the MA takes the average of a certain time window, the SG works by applying a least squares polynomial fit to a moving window of data points over time. The SG method has an advantage over the MA as it better preserves features of the original signal, such as peak height and width, which are usually attenuated by the MA method.

Recurrent neural network (RNN) The RNN has been highly successful in their ability to learn dependencies in time series data by applying non-linear transformation and considering both the current input and the previous hidden state. There are multiple types of RNNs such as the standard RNN, long short term memory (LSTM) [45], bidirectional LSTM (BLSTM) [46–48], and gated recurrent unit (GRU) [49]. RNNs have been successfully applied to many time series data tasks including action recognition [50, 51], speech recognition [52], and image captioning [53].

Standard RNNs are known to have difficulty in learning time dependencies for long time steps [54]. They also process inputs in temporal order which makes their outputs tend to be mostly based on previous hidden states. LSTMs have shown capability of learning long and short time dependencies, but still LSTMs also tend to be mostly based on previous hidden states only. BLSTMs [46–48], therefore, were introduced to process each training sequence forwards and backwards in two separate LSTM nets in which they learn from previous as well as next hidden states. This thesis examines using BLSTM as an *online* AF smoothing method in Chapter 4.

2.2.2 Data Filtering

Filtering methods are usually used to denoise time series data that contain statistical noise. Using filters is mostly the same as smoothing. Filtering methods can be used for smoothing and not all smoothing methods can be adopted as filters. In this section we review the two common filtering methods: (1) Kalman filtering [55] and (2) particle filtering [56].

Kalman filtering Kalman filtering [55], also known as linear quadratic estimation, is a method used to process time series data that suffer from statistical noise and other inaccuracies. This method produces estimates of unknown variables that tend to be more

accurate than those original ones, by estimating a joint probability distribution over the variables for each data point.

The Kalman filtering algorithm works in two phases: the prediction and update phases. In the prediction phase, the Kalman filter estimates the current state variables, along with their uncertainties. Once the algorithm observes the next data point (necessarily to be distorted with some amount of error, including random noise), these estimates are then updated using a weighted average in the update phase. The update is done in a way that larger weights are given to estimates with higher certainty [55].

Particle filtering Particle filtering [56] uses a set of data points (also known as particles) to represent the posterior distribution of some stochastic process given noisy and/or partially observed data points. Particle filtering can handle nonlinear models and does not assume any initial states and/or noise distributions. Particle filter methods are also known for generating samples from the required distribution without requiring any assumptions about the state-space model or the state distributions [57, 58]. However, these methods do not perform well when applied for data of high-dimensional space.

Particle filters perform predictions and updates in an approximate manner. The samples from the distribution are represented by a set of particles; each particle has a likelihood weight assigned to it that represents the probability of that particle being sampled from the probability density function. Particle filters require a re-sampling step to replace the particles with very small weights. There are many adaptive re-sampling algorithms have been introduced, including the variance of the weights and the relative entropy with respect to the uniform distribution [59]. In the re-sampling step, the particles are replaced by new particles in the proximity of the particles that have higher weights.

Unlike Kalman filters, particle filters perform well for systems that do not fit into a linear model, and/or for samples that do not look very Gaussian. Kalman filters are linear quadratic estimators—i.e. they are best for estimating linear systems with Gaussian noise. Kalman filters also have much lower computational complexity compared to particle filters, but are less flexible.

In our work, the lens motion data generated for conventional methods (which we explain in more details in Chapter 3) do not have any random or Gaussian noise added. With that said, the smoothing methods (i.e., MA and SG) are more suitable to smooth such noise-free data. Therefore, we use the output of both the MA and SG methods as a

proxy to the ground truth for training our BLSTM.

2.3 Defocus Deblurring

In Part III, we address DoF manipulation after effect that includes extending DoF (i.e., defocus deblurring) in Chapter 5–7. To this end, we review related defocus deblurring methods in this section.

Related methods in the literature can be categorized into: (1) defocus detection methods [60–65] or (2) defocus map estimation and deblurring methods [66–70]. While defocus detection is relevant to our problem, we focus on the latter category as these methods share the goal of ultimately producing a sharp deblurred result.

Reducing defocus blur is challenging due to the nature of the spatially varying point spread functions (PSFs) that vary with scene depth [71,72]. Most of the existing DoF blur reduction methods [66–70] approach the problem in two stages: (1) estimate the defocus map of the input and (2) apply off-the-shelf non-blind deconvolution (e.g., [73,74]) guided by the estimated defocus map. Representative works include Karaali et al. [67], which uses image gradients to calculate the blur amount difference between the original image edges and their re-blurred ones. Park et al. [69] introduced a method based on hand-crafted and deep features that were extracted from a pre-trained blur classification network. The combined feature vector was fed to a regression network to estimate the blur amount on edges and then later deblur the image. Shi et al. [70] proposed an effective blur feature using a sparse representation and image decomposition to detect just noticeable blur. Methods that directly deblur the image include Andrès et al.'s [66] approach, which uses regression trees to deblur the image. Recent work by Lee et al. [68] introduced a DNN architecture to estimate an image defocus map using a domain adaptation approach. This approach also introduced the first large-scale dataset for DNN-based training.

The performance of the previously mentioned defocus debluring methods [66–70] is bounded by the defocus map estimation and the effectiveness of the non-blind deconvolution. Additionally, due to the two-stage approach, these methods have a very large processing time.

Unlike prior work, we are the first to propose an end-to-end DNN framework for DoF blur reduction in a single stage (Chapter 5). We are also the first to utilize the DP sensor information available at capture time. Our proposed method, in Chapter 5, employs

a DNN that uses the DP sensor's two sub-aperture views as input to predict a single deblurred image. The effectiveness of our method is attributed to the DNN's ability to learn the amount of spatially varying defocus blur from the two DP views. This idea stems from the way the DP sensors work. DP sensors were developed as a means to improve the camera's AF system. The DP design produces two sub-aperture views of the scene that exhibit differences in phase that are correlated to the amount of defocus blur.

2.4 Datasets

Having an appropriate dataset is crucial to develop methods for DoF adjustment. However, there is a lack of real datasets targeting research on video AF and defocus deblurring. Therefore, in this section, we review datasets related to camera AF (Section 2.4.1) and then existing defocus deblurring datasets (Section 2.4.2).

2.4.1 Focal Stack Datasets

Beyond various ad-hoc focal stack data available online from class projects and photography enthusiasts, there are very few formal focal stack datasets available for academic research. Two notable datasets are by Mousnier et al. [75] and Li et al. [76]. The dataset in [75] provides 30 focal stacks of static scenes of images of size 1088×1088 pixels. The dataset in [76] captured 100 focal stacks of image size 1080×1080 pixels, again of static scenes. The number of images per focal stack ranges from 5 to 12. These datasets are not intended for the purpose of AF research, but instead target tangentially related topics, such as digital refocusing [77–79], depth from defocus [80, 81], and depth from focal stacks [82].

In addition, the focal stacks in these datasets are synthetically generated based on the Lytro light field camera [83, 84]. Unfortunately, the consumer-level Lytro devices do not support video capture. The new Lytro Cinema does offer video light field capture, but the cost of renting this device is prohibitively high (in the hundreds of thousands of dollars). Moreover, the Lytro Cinema is not representative of smartphones. Unlike the datasets in [75, 76], in this thesis (Chapter 3), we introduce a dataset for video AF purposes that provides a much larger focal stack of 50 images of size 3264×1836 pixels. Our dataset consists of 10 temporal image sequences with up to 90 full focal stacks per sequence, and

offers a real optical defocus blur that has been generated naturally by moving camera's lens.

2.4.2 Defocus Blur Datasets

There are several datasets available for defocus deblurring. The CUHK [61] and DUT [64] datasets have been used for blur detection and provide real images with their corresponding binary masks of blur/sharp regions. The SYNDOF [68] dataset provided data for defocus map estimation, in which their defocus blur is synthesized based on a given depth map of pinhole image datasets. The datasets of [61, 64, 68] do not provide the corresponding ground truth all-in-focus image. The RTF [66] dataset provided light-field images captured by a Lytro camera for the task of defocus deblurring. In their data, each blurred image has a corresponding all-in-focus image. However, the RTF dataset is small, with only 22 image pairs. While there are other similar and much larger light-field datasets [85, 86], these datasets were introduced for different tasks (i.e., depth from focus and synthesizing a 4D RGBD light field), which are different from the main goal of this thesis. In general, the images captured by Lytro cameras are not representative of DSLR and smartphone cameras, because they apply synthetic defocus blur, and have a relatively small spatial resolution [87].

As our approach is to utilize the DP data for defocus deblurring, we found it necessary to capture a new dataset. In Chapter 5, we introduce our DP defocus blur dataset that provides 500 pairs of images of unrepeated scenes; each pair has a defocus blurred image with its corresponding sharp image. The two DP views of the blurred image are also provided, resulting in a total of 2000 images. Details of our dataset capture are provided in Section 5.3. Similar to the patch-wise training approach followed in [68, 69], we extract a large number of image patches from our dataset to train our DNN.

Part II

Pre-Capture DoF Manipulation

Chapter 3

AF Dataset and Platform

Camera AF is the process of determining how to move a camera's lens such that certain scene content is in focus. The underlying algorithms used by AF systems, such as PDAF (Section 2.1.2) and CDAF (Section 2.1.3), are well established. However, determining a high-level objective regarding how to best focus a particular scene is less clear. This is evident in part by the fact that different cameras employ different AF criteria; for example, some attempt to keep items in the center in focus, others give priority to faces while others maximize the sharpness of the entire scene. The fact that different objectives exist raises the research question of whether there is a preferred objective. This becomes more interesting when AF is applied to videos of dynamic scenes. This chapter aims to revisit AF for cameras within the context of temporal image data. As part of this effort, we describe the capture of a new 4D dataset that provides access to a full focal stack at each time point in a temporal sequence. Based on this dataset, we have developed a platform and associated application programming interface (API) that mimic real AF systems, restricting lens motion within the constraints of a dynamic environment and frame capture. Using our platform we evaluated several high-level focusing objectives and found interesting insight into what users prefer.

3.1 Introduction

One of the crucial steps in image capture is determining what part of the scene to focus on. In this chapter, we examine this problem for smartphone cameras because smartphones now represent the dominant modality of video and image capture performed by consumers.

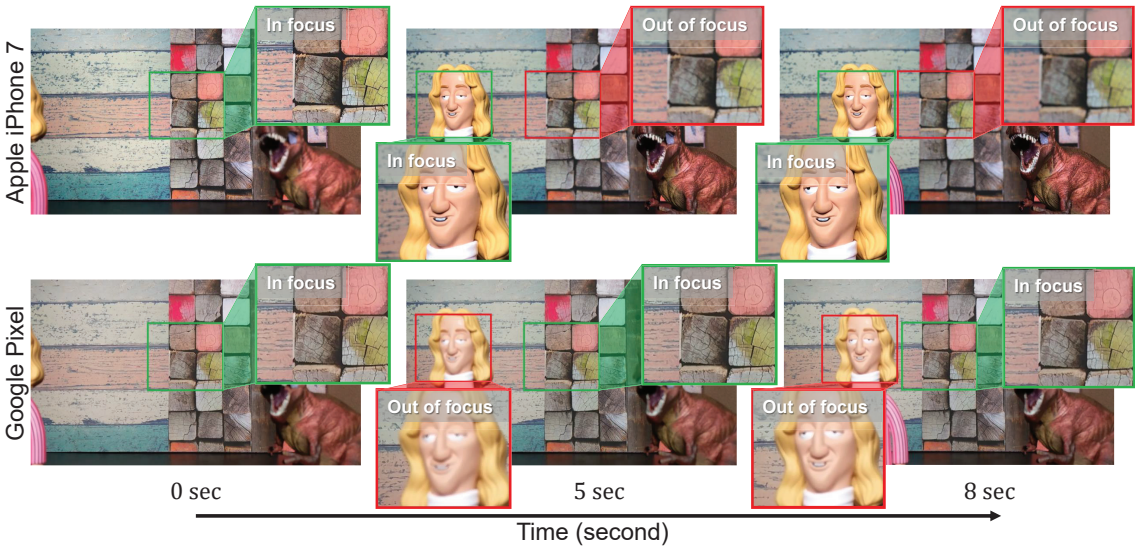


Figure 3.1: An Apple iPhone 7 (top row) and Google Pixel (bottom row) are used to capture the same dynamic scene controlled via translating stages. At different time slots in the captured video, denoted as 0 sec, 5 sec, 8 sec, it is clear that each phone is using a different AF objective. It is unclear which is the preferred AF objective. This is a challenging question to answer as it is very difficult to access a full (and repeatable) solution space for a given scene.

While manual focus is possible on smartphones–either through direct manipulation of the lens position or by clicking on regions of interest in the scene, most users rely on the camera's AF mechanism.

The goal of AF is straightforward. Given some high-level objective of what scene content or image region is desired to be in focus, AF systems attempt to move the lens such that these regions appear sharpest. From an optical point of view, the sharpness correlates to the desired image region lying within the len's depth of field. Smartphone cameras, as opposed to digital single-lens reflex (DSLR) and point-and-shoot cameras, are unique in this regard, since they have fixed apertures and depth of field is therefore restricted to lens position only.

The low-level algorithms used to determine image sharpness–for example, contrast detection and phase differencing–are well established. What is more challenging is using these low-level algorithms to realize high-level AF objectives for dynamic scene content in a temporal image sequence (i.e., video). This is evident from the variety of different AF

criteria used by different smartphone cameras. Figure 3.1 shows an illustrative example. In this example, an Apple iPhone 7 and a Google Pixel have captured a scene with objects that move on a translating stage. The translating stage and controlled environment allow each camera to image the same dynamic scene content. We can see that each camera is focusing on different image regions at the same time slots in the video.

This begs the question of which of these two approaches is preferred by a user. From a research point of view, one of the major challenges when developing AF algorithms is the inability to examine the full solution space since only a fixed focal position can be captured at each time instance. While it is possible to capture a full focal stack for a static scene, it is currently not possible for a temporal image sequence in a dynamic environment. Moreover, there are additional constraints in an AF system beyond determining the right focal position given a full focal stack. For example, the lens cannot be instantaneously moved to the correct focal position; it can only advance either forward or backward within some fixed amount of time, and within this time quantum the scene content may change and the current video frame may advance. This lack of access to (1) temporal focal stack data and (2) an AF platform that holistically incorporates lens motion, scene dynamics, and frame advancement is the impetus for our work.

Contribution The contribution of this chapter is a software platform for AF research and an associated 4D temporal focal stack dataset. Our AF platform allows the design, testing, and comparison of AF algorithms in a reproducible manner. Our focal stack dataset is composed of 33,000 full-frame images consisting of 10 temporal image sequences, each containing 50–90 full focal stacks. Our software platform provides an AF application programming interface (API) that mimics the real-time constraints, including lens motion timing with respect to scene motion and frame advancement. Additionally, we have performed analysis on several smartphone AF algorithms to come up with a set of representative high-level AF objectives. Using our platform and data we have implemented these algorithms to produce similar outputs found on real phones and used the results to perform a user study to see if there are any preferences. Our user study reveals that overall lens motion, and not necessarily the actual scene content in focus, is the predominant factor dictating preference. We believe our dataset and software platform will provide further opportunities for revisiting video AF research. We published the contributions of this chapter in the European Conference on Computer Vision (ECCV), 2018 [14].

Figure 3.2: A top view of our capture environment. Each shot contains the scene components: linear stage actuators, smartphone camera, tripod, objects, and scene background.

Given the nature of this chapter that contains video results, we have organized them into an HTML page file instead of a PDF. Please find chapter-related links below:

- Project page:
  `https://abuolaim.nowaty.com/eccv_2018_autofocus/`
- Video AF dtaset:
  `https://abuolaim.nowaty.com/eccv_2018_autofocus/dataset.html`

3.2 AF Analysis and Dataset Capture

3.2.1 Capture Environment

To begin this work, we constructed an environment that allowed scenes with different content and moving objects to be imaged in a repeatable manner. All videos and images were captured indoors using a direct current (DC) light source to avoid the flickering effect of alternating current lights [88]. To control scene motion, we used three DIY-CNC linear stage actuators that were controlled by a ST-4045-A1 motor driver and Arduino/Genuino Uno microcontroller. Each linear stage has a travel length of 410mm and uses a stepper motor of Nema 23 24V 3A 1N.M. The three linear stage actuators can be combined together to give more degrees of freedom. We calibrated our motors to allow 106 equal steps of 3.87mm each with a motion speed of 9.35mm/s.

3.2.2 Analysis of Smartphones AF

Within this environment, we analyzed the performance of three representative consumer smartphones (Apple iPhone 7, Google Pixel, Samsung Galaxy S6). The manufacturers

Table 3.1: The 10 scenes/image sequences in our AF dataset. See Section 3.2.3 for detail of the table and video/image sequence description. There are three scene categories: no face (NF), face in foreground (FF), and face in background (FB). The final table row, discrete time points, denotes the number of full focal stacks per captured temporal image sequence.

Scene	1	2	3	4	5	6	7	8	9	10
Example image										
Category	NF		FF		NF	FF	FB		NF	FF
Camera	stationary				moving	stationary		moving	stationary	
Camera motion type	—				zig-zag	—		zig-zag	—	
Textured background	✓	✗	✓							
Face	✗		✓		✗	✓			✗	✓
Motion switches	1				3	0	2		2	
Scene motion angle w.r.t image plane	90°		0°, 90°	0°	36°	0°, 52°, 142°	47°, 36°		zig-zag	
Video length	21.6 sec				27.5 sec	29 sec	30.8 sec		39.1 sec	
Discrete time points	51				61	71			91	

of those smartphones do not supply documentation of how they perform AF. However, one of the alternatives is to try to reverse engineer their AF performance would be the controlled testing to observe their AF behaviour under different scenarios. The cameras are positioned such that their fields of view are as similar as possible. The frame rate for video capture is fixed at 30 frames/sec. Given the different optical systems and image formats among the cameras, there are slight differences in the field of view, but these differences are negligible in terms of their effect on the AF outcomes.

We experimented with a wide range of scene configurations, such as an object with a figurine with a human face, textured backgrounds, and various moving objects. As previously illustrated in Figure 3.1, we observed that the AF behaviour differs between different smartphones. For example, in one experiment we set up a textured background

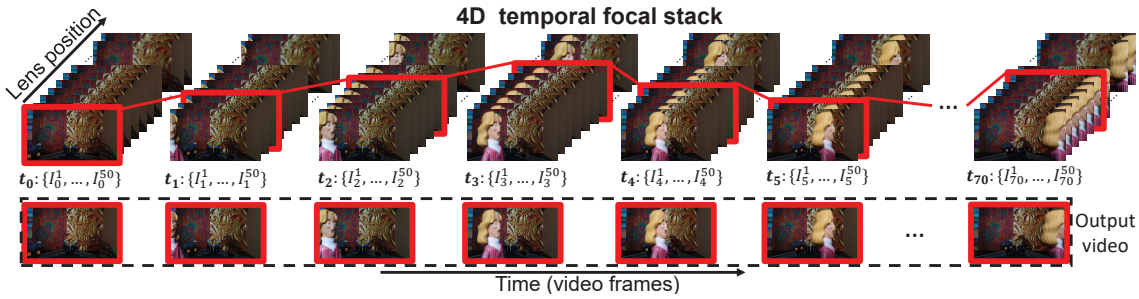


Figure 3.3: This figure shows an example of our 4D temporal focal stack data. The highlighted frames indicate the images that would be visible in the output video. The position of the highlighted images is achieved by moving the lens of the camera.

and a textured object to move horizontally from left to right with respect to the camera. We observed that for the Google Pixel and Samsung Galaxy S6 Edge smartphones, the foreground object becomes in focus only when it is inside the center of the image; otherwise it is out of focus. For the same setup captured by an Apple iPhone 7, however, the foreground object is in focus most of the time regardless of its position from the center. In another experiment with a figurine with a human face, we observed that the three smartphones detected the face in a video, but only Apple iPhone 7 focused on the face.

3.2.3 Scene and Image Sequence Capture

Based on our observations, we settled on 10 representative scenes that are categorized into three types: (1) scenes containing no face (NF), (2) scenes with a face in the foreground (FF), and (3) scenes with faces in the background (FB). For each of these scenes, we allowed different arrangements in terms of textured backgrounds, whether the camera moves, scene dynamics including motion angle with respect to (w.r.t) the image plane (i.e., parallel, perpendicular, and/or angular), and how many types of objects in the scene that change their direction (referred to as *motion switches*). Table 3.1 summarizes this information. Figure 3.2 also shows the physical setup of several scenes.

For each of these 10 scenes, we captured the following data. First, each scene was imaged with the three smartphone cameras. This video capture helps to establish high-level AF objectives used on phones and determines the approximate video length needed to capture the overall scene dynamics. The duration of these videos is provided in Table 3.1.

Example videos captured by different smartphones are provided in the project page[1].

Next, we captured temporal focal stacks for each of these scenes. We refer to these as *image sequences* to distinguish them from the actual videos. To capture each image sequence, we replicated the video capture in a stop-motion manner. Specifically, the objects in the scene are moved in motion increments of 3.87mm between consecutive *time points*. We used the Samsung Galaxy S6 Edge smartphone to perform the image capture using a custom Android app that fixed all camera settings (e.g., ISO, white balance, shutter speed). Our app also controlled the lens position, such that for each *time point* t_i, we captured a focal stack of 50 images where the camera lens is moved in linear steps from its minimum to maximum position. The last row in Table 3.1 shows also the number of *time points* for each captured temporal image sequence. In this thesis, we use the term *time point* to denote a time slot in our stop-motion data. We also use the term *frame* to denote a real-time video frame, either from a real video or an output produced by our AF platform.

The Samsung Galaxy S6 Edge is an Android-based smartphone that allowed us to have stable control of lens motion through the Camera2API interface with the Android/JAVA code. For the iOS-based smartphone, there was no API that allowed us to access low-level camera API controls like moving the lens. There is another Android-based smartphone (i.e., Google Pixel) we used for testing and analysis, but that one showed unstable control of lens movements using the Camera2API interface (e.g., abrupt change of focus distance and camera exposure) that made it hard for us to use it for data capture. With that said, The Samsung Galaxy S6 Edge smartphone is the only device used for dataset collection, whereas the other devices (i.e., Apple iPhone 7 and Google Pixel) are used for analyzing and testing the camera's AF behavior.

The images captured by the Samsung Galaxy S6 Edge smartphone are in the sRGB color space. Capturing RAW data with the Camera2API interface showed unstable control of lens movements as having stable control of lens motion is crucial for this task. While processing images in the RAW space has an advantage for tasks like camera AF that happen early in the camera pipeline, hardware constraints prevented us from capturing RAW data with stable lens control.

Figure 3.3 shows an example of scene 6 with 71 *time points*. Each *time point* t_i in

[1] `https://abuolaim.nowaty.com/eccv_2018_autofocus/supplemental_materials/supplemental_materials.html`

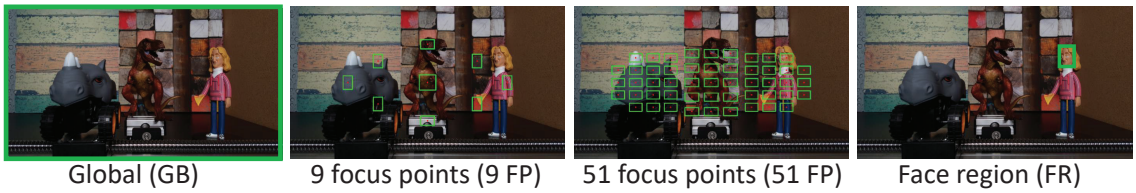


Figure 3.4: Our four AF objectives. The region bounded in a green box is a candidate for RoI(s).

Figure 3.3 has a focal stack of 50 images that are denoted as I_i^j, $j = 1, ..., 50$, where i denotes time point and j indexes the focal stack image associated to a specific lens position.

3.3 AF Platform and API

We begin with a short discussion on how our platform emulates PDAF and CDAF as these are the low-level algorithms of any AF system. This is followed by a discussion on the overall platform and associated API.

3.3.1 PDAF/CDAF Emulation

The CDAF and PDAF processes can be divided into three main steps: first, determine a desired region of interest (RoI) based on the high-level AF objective; second, measure the sharpness or phase of the RoI selected; third, adjust the lens position to maximize the focus.

Based on the observed behaviour of the captured video from our three smartphone cameras on the 10 scenes, we determine four high-level AF objectives in terms of RoI as follows: (1) global (GB) RoI targeting the whole image; (2) a layout of 9 focus points (9 FP) with 9 RoIs; (3) a layout of 51 focus points (51 FP) with 51 RoIs; (4) and a *face region* (FR) RoI where the largest region of detected faces is set as the RoI. Figure 3.4 shows the RoI(s) for each objective bounded in a green box.

Our AF platform provides the flexibility to manually specify the RoI; however, based on the above four objectives, we provide these as presets that the user can select. To facilitate the face region objective for our dataset, we manually labeled the face regions to avoid any face detection algorithm mistakes. Our platform allows retrieval of the labeled face region via an API call; however, when the pre-defined face region is selected, this call

Table 3.2: API calls with their parameters and return values. Each API call incurs a cost related to the number of internal clock cycles. C_{loc} current clock cycle, C_{glob} current time point, $I^{j}_{C_{\text{glob}}}$ current image at current C_{glob} and current lens position j, o optimal lens position, and *score* is the score of gradient energy (default or defined by user).

API call	Description	Return values	Clock cycles
`setScene(int sc)`	Select one of the 10 scenes, `sc`= $0, ..., 9$	`null`	0
`setRegion(int [] reg)`	Set the region either by selecting one of the predefined regions: Global (`reg`=[0]), 9 Focus Points (`reg`=[1]), 51 Focus Points (`reg`=[2]) or Face Region (`reg`=[3]), or by passing an array of size $r \times 4$ where r is the number of regions. Each region has offset (x,y), width, and height.	`null`	0
`setSharpMeasure(int sh)`	Select one of the two predefined sharpness measures: Sobel (`sh`=0) or Prewitt (`sh`=1).	`null`	0
`setKernelSize(int ker)`	Select one of the three predefined kernel sizes: 3 (`ker`=0), 5 (`ker`=1) or 7 (`ker`=2).	`null`	0
`recordScript()`	Start recording the subsequent API calls in a script.	`null`	0
`endScript()`	Stop recording the subsequent API calls in a script.	`null`	0
`callPD(int` ρ`)`	Compute phase difference and return approximate optimal lens position $o \pm \rho$.	$[C_{\text{loc}}, C_{\text{glob}}, I^{j}_{C_{\text{glob}}}, j, o]$	1
`callCD(function fun)`	Allow the user to pass custom contrast detection AF implementation as a function. Default Sobel/Prewitt with kernel size as set by user. `fun` is a function written in Python format.	$[C_{\text{loc}}, C_{\text{glob}}, I^{j}_{C_{\text{glob}}}, j, score]$	1 (if default) or defined by user
`moveLensForward()`	Move the lens a step forward.	$[C_{\text{loc}}, C_{\text{glob}}, I^{j}_{C_{\text{glob}}}, j]$	1
`moveLensBackward()`	Move the lens a step backward.	$[C_{\text{loc}}, C_{\text{glob}}, I^{j}_{C_{\text{glob}}}, j]$	1
`noOp()`	No operation. No lens movements. Used to increment C_{loc} in order to move in global time C_{glob}.	$[C_{\text{loc}}, C_{\text{glob}}, I^{j}_{C_{\text{glob}}}]$	1
`getFaceRegion()`	Detect face(s) and return face region(s) int `face[]` if exists. `face[]` is an array of face regions found in $I^{j}_{C_{\text{glob}}}$. Each face region has an offset (x,y), width, and height.	$[C_{\text{loc}}, C_{\text{glob}}, I^{j}_{C_{\text{glob}}}, \texttt{face[]}]$	0

is automatically performed and the RoI is set to the face region. Regarding the sharpness measure for the CDAF, we provide two gradient-based filters–namely, Sobel and Prewitt operators. Based on Loren's findings in [29], the Sobel and Prewitt filters are the most accurate among other sharpness measure methods. The size of these filters can also be controlled.

3.3.2 AF Platform and API Calls

Our AF API is designed to emulate AF in smartphones. The platform and API impose constraints on lens motion timing with respect to scene motion and video frame rate. As such, our API and platform have a *local* and *global* virtual clock. The local clock, denoted

as C_{loc}, emulates the real-time internal clock on the smartphone, whereas the global clock, C_{glob}, emulates the real-world timing (scene dynamics).

Platform timing Since the Samsung Galaxy S6 smartphone was used to capture our dataset, we measured its performance to establish the mapping between the local and global clocks. Specifically, we measured how long it took the camera to respond to a scene change at a different focal positioning by sweeping the lens to this position while capturing video. To do this, we set up two objects: a textured flat background and a textured flat foreground; both are parallel to the camera plane at different depth layers (one close and one far). The background object appears at the beginning of the video capturing and is in focus; then, after a short delay we immediately display the foreground object closer to the camera, which causes the AF system to move the focus from background to foreground. Later, we decompose the captured video into frames and count how many frames it required to move from background to foreground. For the exact same scene scenario, we collected a full focal stack (50 images), previously discussed. To obtain how many steps the lens moved, we use the focal stack to compute at which lens positions were the background and foreground objects in focus.

Once we obtain the number of lens steps and number of frames required, we can compute from lens step to frame unit (33.33 msec). Therefore, we estimated the Samsung Galaxy S6 Edge requires 42 msec to move the lens one step (including image capturing and AF processing).

The time required for the translating stage motor to move one step (3.87mm) is 414 msec. Recall that a single translating stage motor step in real time is equivalent to a discrete time point in our stop-motion setup. Therefore, the number of steps allowed for the lens to move in one time point is equal to $414/42 \approx 9.86$ steps. Based on this approximate calculation, we fix it to 10 steps and we relate it to the local clock C_{loc} (one lens movement costs one clock cycle). Accordingly, the corresponding global clock C_{glob} increments every 10 clock cycles. Thus our relationship is: 10 C_{loc} advances C_{glob} by 1.

API Our API is based on Python and provides 12 primitive calls as described in Table 3.2.

To prepare the API for receiving user calls, the user needs to run the API software which operates like a network server. The API will return the string "start" indicating

that the API has started and is ready to receive user program calls.

The following API calls are used to set up the API environment. These calls all return `null` and cost 0 cycles:

- `setScene(int sc)`: The user can pass the scene number (`sc`$= 0, ..., 9$) to select one of the ten scenes. We upload the entire temporal sequence at the beginning of this call as a single 4D array. This is loaded into the main memory to speed up image access. Uploading the scene takes from 7 to 15 sec based on scene size ($n \times 50$ images).
- `setRegion(int [] reg)` (see description in Table 3.2).
- `setSharpMeasure(int sh)`(see description in Table 3.2).
- `setKernelSize(int ker)`(see description in Table 3.2).

The `recordScript()` and `endScript()` API calls are used to record other API calls and associated metadata to an output script. This output script can be loaded later for user playback purposes. Both calls return `null` and cost 0 cycles. `recordScript()` creates a output script named by the user and writes user info with all API environment parameters as comments. Then, it starts recording all subsequent API calls. `endScript()` writes a summary and metadata about the performance–for example, total number of lens movements, lens position, API call made at each clock cycle, etc–as comments, too. Additionally, It stops recording API calls and closes the script.

Our callPD(int ρ) API call is used to emulate the phase difference autofocus (PDAF) available on most high-end smartphone cameras. The *real* PDAF routine on a camera is able to find the approximate lens position for a desired RoI close to the optimal focal frame in the focal stack within a single processing pass of the low-level raw image. On real cameras, the PDAF result is obtained at a hardware level based on a proprietary layout of dual-pixel diodes placed on the sensor. We were not able to access this data and provide it as part of our focal stack dataset. As a result, we instead emulate the result of the phase difference by running CDAF targeted to the specified RoI on the whole focal stack at the current time-point t_i defined by the global clock C_{glob}. As mentioned previously, real camera PDAF is performed first to move the lens closer to the optimal focusing position; afterwards CDAF is typically performed to refine the lens position. To mimic this near

optimality, we apply an inaccuracy tolerance ρ on the optimal focusing position obtained. This inaccuracy tolerance allows the estimated lens position to lie randomly around the optimal by $\pm[0, \rho]$ and is a parameter that can be passed to the API.

The `callCD(function fun)` is added to mimic the contrast detection autofocus (CDAF) and to allow future algorithm development, scalability and give more flexibility to user implementation. By allowing this feature, our API CDAF can be easily extended. The user would need to develop his own CD implementation, define his own parameters and return values. The CDAF implementation, thus, can be passed as a function `fun` written in Python. Then, the API is responsible to implement user-custom CDAF implementation. The `callCD(function fun)` will take default values if the user passes nothing and return the gradient magnitude energy score *score* of the RoI(s) based on the default objective. For example, if the objective is 9 focus points, the *score* will be an array of size 9 where each index represents the gradient energy of a specific region. Both `callPD(int` ρ`)` and `callCD(function fun)` cost 1 clock cycle with default implementation. However, `callCD(function fun)` may cost more if it is computationally expensive and this can be defined by the user.

Regarding `noOp()` API call. This call does nothing except that it advances the 1 clock cycle to increment the internal clock C_{loc}. This is an important call used to fill clock cycles C_{loc} (increment till 10 cycles) in order to advance the global clock C_{glob}. For example, suppose that the lens position is optimal at current C_{glob} and the user does not want to move the lens but still he needs to access the next time point (C_{glob} +1). The only thing possible is to keep calling `noOp()` to increment C_{loc} in order to advance the clock (C_{glob} will be automatically incremented every 10 clock cycles).

3.3.3 Example Implementation

Alg. 1 provides a simple pseudo-code based on our API to demonstrate how an AF algorithm based on the *global objective* for Scene 4 can be implemented. Real Python examples and script recording and video outputs are provided in the project page[2]. In this simple example, we set the ρ to zero, which results in `callPD()` calls returning the optimal lens position.

[2] `https://abuolaim.nowaty.com/eccv_2018_autofocus/supplemental_materials/supplemental_materials.html`

Algorithm 1 Example of a Global RoI Objective using Optimal PDAF

```
1: Start API
2: setScene(Scene4)
3: setRegion(Global)
4: recordScript()                          //Create a script and start recording API calls
5: while not end of time points do
6:     if time point t_i incremented then
7:         C_loc, C_glob, I_i^j, j, o ←callPD(0)
8:     else if optimal lens position o > current lens position j then
9:         C_loc, C_glob, I_i^j, j ←moveLensForward()
10:    else if optimal lens position o < current lens position j then
11:        C_loc, C_glob, I_i^j, j ←moveLensBackward()
12:    else if optimal lens position o == current lens position j then
13:        C_loc, C_glob, I_i^j ←noOp()
14:    end if
15:    Video ← I_i^j                       //Write the acquired image into a video
16: end while
17: endScript()          //Close and summarize the script(e.g., # of lens movements)
```

Based on our implementation in Alg. 1, the time point t_i will be incremented by the API every 10 clock cycles (as discussed before in Section 3.3.2). At each clock cycle, the API returns an image, which means 10 images are obtained at each t_i. Thus, the total number of images returned by the API for a specific scene is equal to 10 multiplied by the scene size in *time points*. To generate an output video for a scene, each image is written at each clock cycle out to a video object. Running Alg. 1 will return metadata about the performance of the *global objective* for Scene 4. In Figure 3.5 we show the lens position over local time (clock cycles) for the *global objective* (GB) in the dark blue solid line. Figure 3.5 illustrates the lens movements over time, where the GB has fewer lens movements and less oscillation. Figure 3.5 also shows the lens position over time for other objectives for Scene 4.

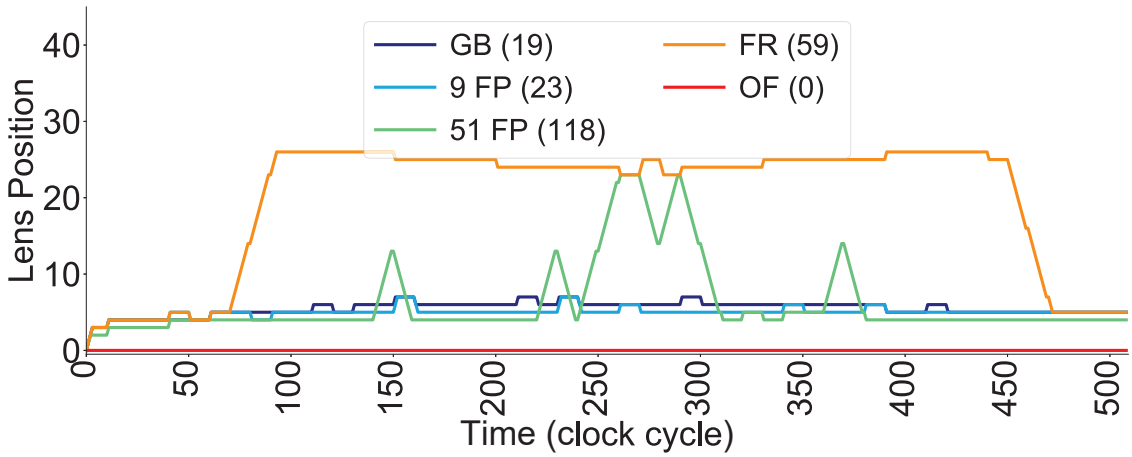


Figure 3.5: This figure shows the lens position over time at each clock cycle for Scene 4 with different objectives. Total number of lens movements is shown in parentheses. An out-of-focus objective (OoF) is included that does not move the lens over the entire sequence. For Scene 4, the 51 focus points (51 FP) objective oscillates the most. For the face region (FR) objective, the face did not enter the scene until clock cycle 70–the 9 focus points (9 FP) are applied by default when the face is not present. Global (GB) and 9 FP objectives tend to oscillate less than others with fewer lens movements.

3.4 User Study on AF Preference

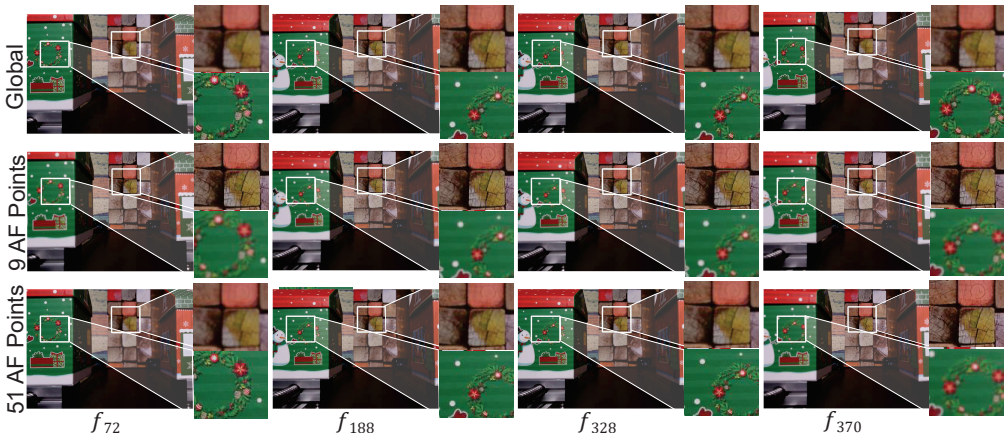
A user study was conducted to determine if there was any particular user preference for the different AF methods. As shown in Figure 3.5, the AF platform gave an opportunity to track exact lens movement for each method. Lens motion was treated as a potential factor.

Preparation For this study we defined scene number, objective, and lens motion as our independent variables; the user preference is our dependent variable. We adopted a force-choice paired comparison approach that requires each participant in the study to choose a preferred video from a pair of videos. Both videos in a given pair are of the same scene but have different AF objectives. We used all 10 scenes from our dataset for the study. There are six scenes with faces and four without. For the scenes with faces, there are four AF objectives–namely, global, 9 focus point, 51 focus point, and face region. The scenes without faces have only the first three AF objectives.

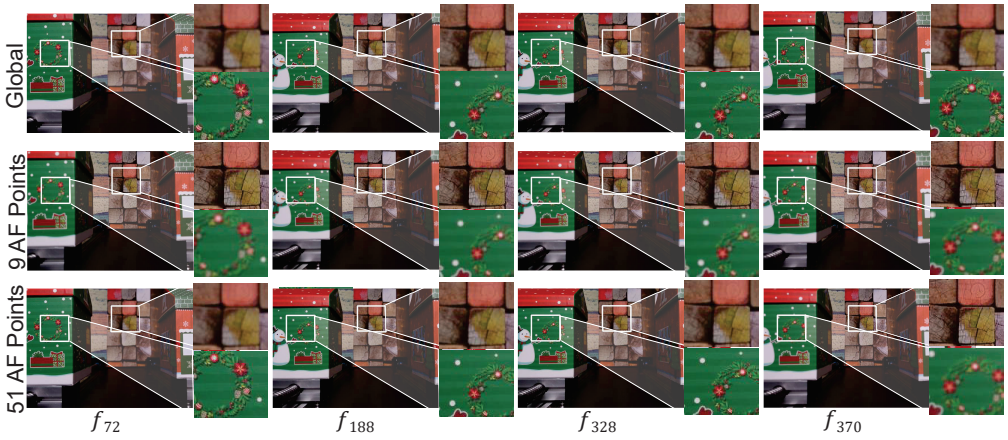
Output videos were generated through our API and using our dataset and modifications of Alg. 1 for each AF objective on all scenes (example video frames from Scene 1 and Scene 10 are shown in Figure 3.6). Additionally, for each scene, we have generated an out-of-focus video, where all scene elements are out of focus. Those out-of-focus videos are generated through our API by fixing the lens to the maximum position and calling `noOp()` till the end-of-scene time points. However, for Scene 6, we omitted this objective because there is no lens position that makes all scene elements out-of-focus. Therefore, there are five scenes in total with five AF objectives (with the out-of-focus objective added), and another five scenes with only four AF objectives. The total number of paired comparisons is $5 \times \binom{5}{2} + 5 \times \binom{4}{2} = 80$.

Procedure We collected 10 opinions for each video pair from 80 participants (34 females and 46 males) ranging in age from 18 to 50. Each subject was shown 10 video pairs selected in random order. We designed a simple graphical user interface that allows the user to view video pairs, one pair after the other, and easily examine the difference in AF behavior. The interface allows the participants to watch the two videos in the current pair any number of times before they make a selection and proceed to the next pair. The survey takes on average three to five minutes to complete. The experiments were carried out indoors with calibrated monitors and controlled lighting. All the users participated in this study were naive and had no idea about the purpose of this study. They were only asked to choose the video they prefer (i.e, the force-choice paired comparison approach).

Outcomes As described earlier, the scenes are categorized as Cat. 1: scenes with no face (NF), Cat. 2: scenes with a prominent face in the foreground (FF), and Cat. 3: scenes in which the face is in the background (FB). For each category, we aggregated user votes into an overall score that represents user preference by counting the number of times each AF objective is preferred over any other objective. These results are presented in Figures 3.7. In Figure 3.7, in the first column, the average user preference per AF objective for each category is shown (i.e., aggregated over scenes). It can be seen that for NF videos, the global (GB) AF objective is the most preferred. For the FF videos, the face region (FR) AF objective is the most preferred. For the FB videos, there is no strong preference among the three objectives GB, 51 focus points (51 FP), and FR, but the most preferred is GB followed by FR. Additionally, we calculated the 95% confidence intervals for these



Scene 1



Scene 10

Figure 3.6: Example output video frames generated by our AF platform using different objectives over time applied on Scene 1 (top) and Scene 10 (bottom).

results as represented by the error bars, which indicate the statistical significance of the results. The results of this user study were statistically significant ($F = 33.5$, $p < 0.0003$). Furthermore, the plots on the right of Figure 3.7 represent the user preference per objective for individual scenes (lower plots) with a corresponding number of lens movements (upper plots with grey bars) for each of the three categories. The individual scene plots also confirm the observations from the aggregate plots for all cases except Scene 9.

To examine the correlation between user preference and the number of lens movements for each category, the user preference vs. lens movements was plotted for each category,

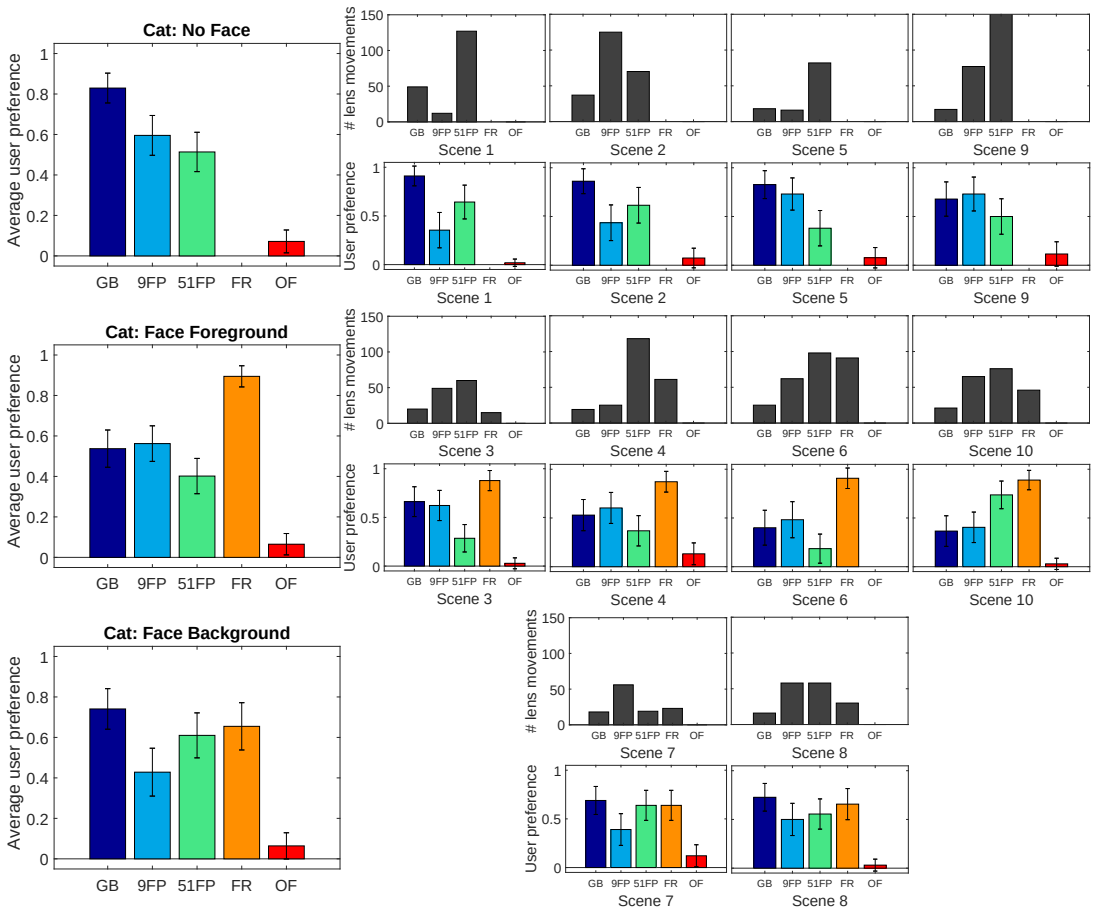


Figure 3.7: User preference of AF objectives for three scene meta-categories: no face (NF), face in foreground (FF), and face in background (FB) for AF objectives: global (GB), 9 focus points (9 FP), 51 focus points (51 FP), face region (FR), and out-of-focus (OoF). The left column shows the average user preference. The small plots on the right show user preference (lower plots) and lens movements (upper plots in grey) for individual scenes.

as shown in Figure 3.8. There is a clear correlation between user preference and lens movements, suggesting that users tend to prefer the objectives with fewer lens movements. This is indicated by the negative correlation coefficients shown on the plots.

For the second category that contains a prominent face in the foreground, the results suggest that users prefer the face AF that locks onto the face even if more motion of the lens is required to achieve this objective. This voting pattern can be seen in the second row in Figure 3.7, where the FR AF objective receives a higher percentage of votes than

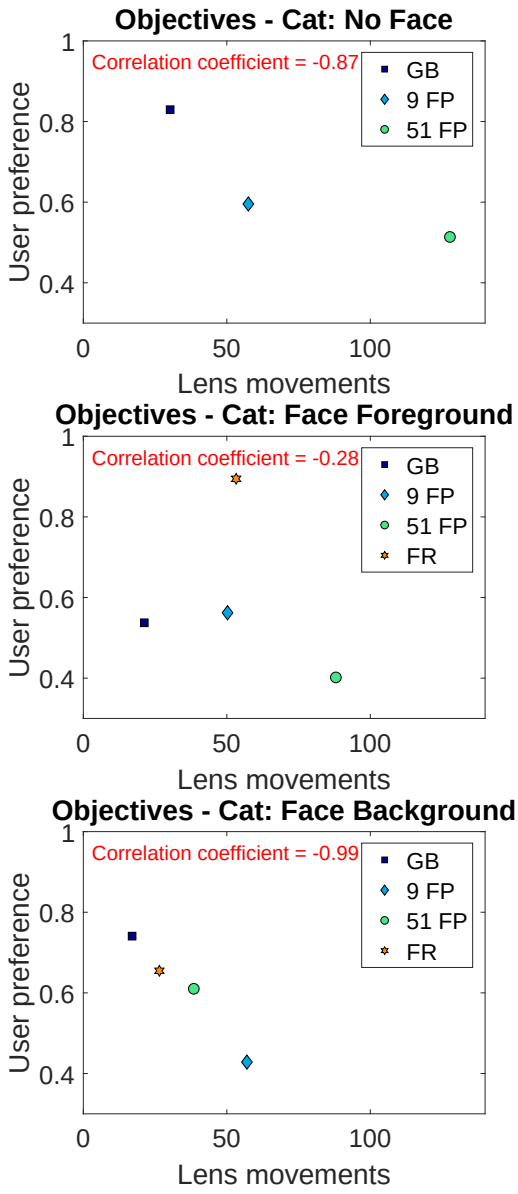


Figure 3.8: The relationship between user preference and number of lens movements for AF objectives for the three scene meta-categories. Top: no face (NF). Middle: face in foreground (FF). Bottom: face in background (FB).

the GB AF, which has the least amount of lens motion. Also note that the 51 focus points (51 FP) objective has the highest amount of lens motion and is the least preferred. In the third category that contains a face in the background, users do not seem to have any strong preference, as seen by the near-equal distribution of votes across 51 FP, GB, and

FR, all of which interestingly have roughly the same amount of lens motion (third row in Figure 3.7). It is also important to note that in agreement with our findings for the first two categories, the objective with the highest amount of lens movement, which in this case is the 9 focus points (9 FP) objective, is the least preferred.

The out-of-focus (OoF) objective is preferred the least across all three categories although it has the least amount of lens motion. This agrees with the common wisdom that at least a part of the scene has to be in focus, and simply minimizing the amount of lens motion does not induce a higher preference.

3.5 Summary

In this chapter, we introduced a new software platform and dataset focused on autofocus for video capture with smartphone cameras. To this aim, we constructed a hardware setup that allows dynamic scenes to be accurately "replayed". Using this environment, we analyzed representative smartphone cameras' AF behaviour under 10 different scenes with various motions, backgrounds, and objects (including an object serving as a proxy for a human face). We also captured these scenes with discrete time points, producing a 4D temporal focal stack dataset for use in AF research. The overall dataset consists of 33,000 smartphone camera images and has been made publicly available. We also developed an AF platform that allows the development of AF algorithms within the content of a working camera system. API calls allow algorithms to simulate lens motion, image access, and low-level functionality, such as PDAF and CDAF. This platform also restricts an AF algorithm to operate within a real camera environment, where lens motion that is directly tied to the systems clock cycle and scene motion is required to access different images in the focal stack.

From our analysis of the cameras' AF behaviour we examined four high-level AF objectives–namely, global, 9 focus points, 51 focus points, and face region. Using our AF platform, we implemented these high-level AF objectives to produce several video outputs that were used in a user study. Because our AF platform allowed accurate analysis of the underlying AF algorithms, we were able to determine that user preference is correlated higher to the overall lens motion than the actual scene objective used. For scenes with faces, focusing on the face (when sufficiently large) took priority, followed by the amount of lens motion. While these findings are somewhat intuitive (e.g., no one likes a scene

with too much lens wobble), as far as we are aware, this is the first study to confirm these preferences in a controlled manner. We believe having access to our temporal focal stack dataset and AF platform is a valuable resource for the research community.

Chapter 4

Lens Motion Smoothing

In the previous chapter, we conducted a user study to investigate users' preference for an AF algorithm. The main finding was that the users' preference was correlated with the videos that had less lens motion and not what part of the scene was in focus (i.e., the RoI used). This finding led to the exploration of reducing lens motion while also considering RoI sharpness. To that end, the aim of this chapter is not to come up with a new AF objective in regards to "what to focus on", but rather an algorithm for temporal smoothing of lens motion in videos based on existing focusing objectives.

We begin this chapter with an introduction (Section 4.1) followed by describing the AF software platform (Section 4.2) and data used in our experiments (Section 4.3). Afterwards, our *online* smoothing algorithms are described in Section 4.4 and Section 4.5, respectively. This material is followed by our quantitative and qualitative results to evaluate our algorithms (Section 4.6). Lastly, we present our second user study to determine which AF videos are preferred (Section 4.8).

4.1 Introduction

When capturing a single image, the goal of AF is clear – move the lens to maximize image sharpness for some desired scene content and then capture the image. When capturing a video (i.e., temporal images), the goal is less clear. The challenge for video AF is often attributed to determining when and where to shift the focus while the scene content changes. The obvious solution for video AF – and one used by most digital cameras – is to apply the single image strategy to each incoming frame to maximize the overall image

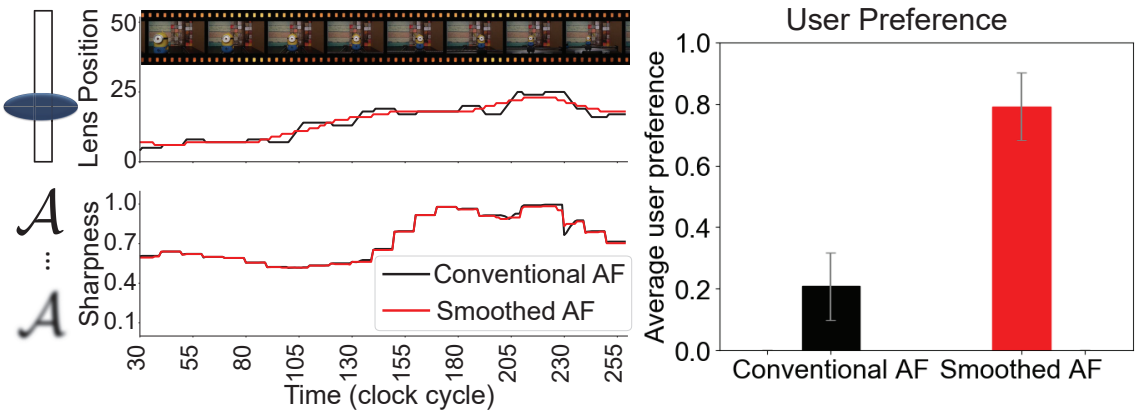


Figure 4.1: The left-hand plot shows lens motion and per-frame sharpness for a conventional AF algorithm and its corresponding smoothed lens movements. The right-hand plot shows the preferences of 32 users for videos using conventional AF and those that have had their lens movement smoothed.

sharpness [14, 38, 40]. However, this approach overlooks the fact that the lens motion can be observed in the captured video. Too much motion can result in undesired 'lens wobble'.

We performed a user study in Chapter 3 to determine what part of the scene users preferred to have in focus — for example, a face region of interest (RoI) if a face was present in the video. We found that user preference *was not correlated* to a particular RoI, but instead correlated to the amount of lens motion in the overall output video. Specifically, users preferred videos that had the least amount of lens motion – irrespective of what RoI was used. The only exception was when a face was present. For such cases, users tolerated large lens movements to bring the face into focus. Moreover, less lens motion not only is preferred by users but also reduces the power consumption required to move the lens [89]. Our user study finding serves as our impetus to explore the idea of smooth lens motion in more detail. Figure 4.1 shows an example.

Contribution We propose two new AF methods for video that incorporate *online* smoothing to reduce overall lens motion. To this end, we generate several lens trajectories for video sequences using conventional algorithms that target maximizing image sharpness. These lens trajectories are then smoothed in an *offline* manner (i.e., smoothing filter uses a window that considers prior and future lens positions) to produce videos with smoothed lens motion. We perform a user study to re-confirm that these smoothed lens trajectories

are preferred over the videos targeting maximum image sharpness. These *offline* smoothed trajectories, however, cannot be performed in a real AF system because they require knowledge of where the lens will move in unseen frames. To perform *online* smoothing, we propose two methods: (1) a supervised bidirectional long short-term memory (BLSTM) module trained on smooth trajectories and (2) an unsupervised weighted moving average (WMA) method that considers scene sharpness and prior lens motion. We demonstrate good quantitative and qualitative results that show our methods can notably reduce lens motion (up to 64%) with a very small loss in sharpness (less than 5.2%). We show that both algorithms produce outputs more preferred than videos produced with no smoothing. Users had a slightly higher preference for the WMA approach. This is an interesting finding as the WMA approach is unsupervised and can be easily incorporated into existing AF camera systems with little computational overhead. The contributions in this chapter were published in the Winter Conference on Applications of Computer Vision (WACV), 2020 [23].

Given the nature of this chapter that contains video results, we have organized them into an HTML page file instead of a PDF. Please find chapter-related links below:

- Project page:
  `https://abuolaim.nowaty.com/wacv_2020_autofocus_lens_motion/`
- WACV presentation:
  `https://www.youtube.com/watch?v=85z075A3rI0`

4.2 Video Focal Stack Dataset

We use our 4D temporal focal stack dataset and AF software platform to generate the output video data (Chapter 3). The dataset contains ten scenes (referred to as Scene 1–10), each consisting of 50–90 full focal stacks of 50 images each, one image for each possible lens position. The AF platform is intended to emulate an AF system on a smartphone. Timing for the AF system is linked to an internal clock cycle, where movement of one lens position requires a single clock cycle. Frame capture is automatically triggered at each clock cycle. After each clock step, the optimal lens position is provided from the PDAF module depending on the AF RoI used. Motion in the scene (i.e., advancement to the next focal stack) is performed after 10 clock cycles (i.e., a lens can move ten times before

moving to the next focal stack). Thus, a scene with 90 focal stacks will be simulated as 30 seconds of video, at an effective frame-rate of 30 frames per second, with approximately 900 lens movements allowed. While this seems low, the motion of objects in the scenes is relatively slow (controlled by linear actuators) and the resulting videos appear visually consistent. Recall, the AF platform API provides four predefined AF objectives in terms of region of interest (RoI)—namely, global RoI targeting the whole image; 9 focus-points (9 FP) with 9 RoIs targeting the center of the frame; 51 focus-points (51 FP) with 51 RoIs; and a face region (FR) RoI.

Using the 4D dataset and modifications of Alg. 1, we generate ten videos for each objective, with the exception of face region, since only six scenes contain faces. In total we have 36 videos (10 for each of the three objectives and 6 for the face region). With each output video, the API also provides meta-data, such as total number of lens movements, lens positions and the optimal (sharpest) lens positions based on PDAF at each clock cycle. Figure 3.3 shows an example of the 4D temporal focal stack data with the frames used to generate the output video.

Figure 4.2 shows a small sample output of the AF platform relevant to our task. At each clock cycle, the optimal lens position from the PDAF module is received, the current lens position, and the change in lens position. Lens motion at each clock cycle can take only one of three values: move forward (+1), move backward (-1), or no movement (0). Figure 4.2-A shows output for the conventional method. Figure 4.2-B shows *offline* smoothing applied on lens motions (Section 4.3). The smoothing is applied as a moving window (i.e., box filter) centered around the current lens position. Such *offline* filtering cannot be realized in a real camera system as it requires knowledge of future lens positions. This *offline* generated information will be used as part of our training data. The goal for an *online* AF system, given the past history of the lens positions and the current optimal lens target from the PDAF module, is to decide whether to move forward, move backward, or not to move at each new clock cycle. This is illustrated in Figure 4.2-C (Section 4.4).

4.3 Lens Smoothing Training Data for Supervised Learning

Before describing our *online* bidirectional long short-term memory (BLSTM) method, we describe the *offline* data smoothing performed to compute the training data. We apply the two common *offline* smoothing methods—namely, moving average (MA) and Savitzky-

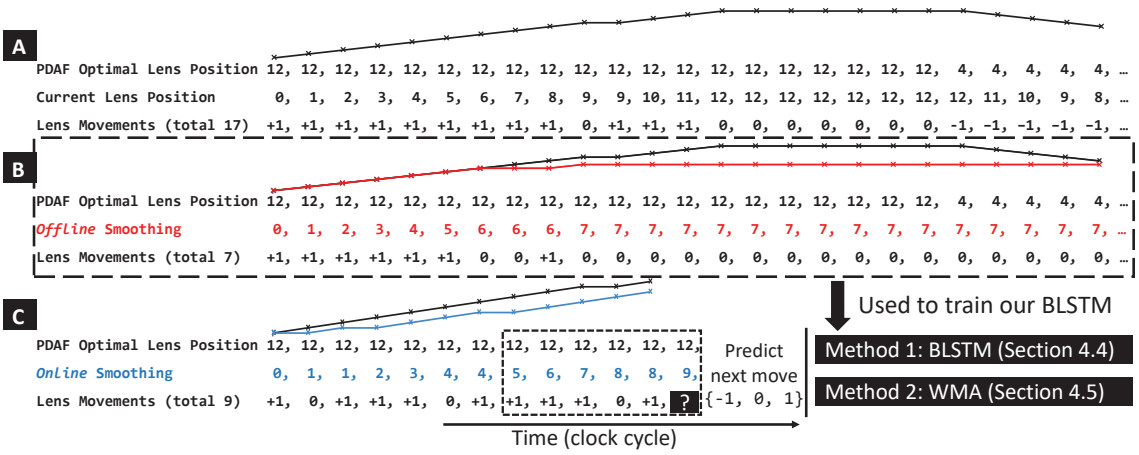


Figure 4.2: This figure provides an overview of the data used for our AF algorithms. (A) shows the output of our AF platform. At each clock cycle we have access to the optimal target lens position from the PDAF. We also have the current lens position as well as the next lens movement. (B) shows the results of applying *offline* smoothing to the actual lens positions. This is shown in red text. A plot of the lens position is also shown. (C) shows the problem of an *online* AF method. The *online* method must predict the next lens move based on prior observations shown in blue. To this end, we propose a learning-based method that uses *offline* smoothed training examples (e.g., from (B)).

Golay (SG). The MA method is the simplest smoothing method in the literature. The idea is simply to compute the average value of each sub-set in successive sub-sets of l adjacent lens positions $x_i, i = 1, ..., l$. These successive sub-sets can be formed by sliding a time window of size l (odd number) and stride of 1. The average value of a certain sub-set centered around x_i is considered as the smoothed lens position x_i^*. Then x_i^* is rounded to the nearest integer to be a valid lens position.

The SG method fits successive sub-sets of l adjacent lens positions $x_i, i = 1, ..., l$ with a low-degree polynomial. The successive sub-sets can be formed by sliding a time window of size l (odd number) and stride of 1. Then the smoothed lens position x_i^* can be computed as follows:

$$x_i^* = \frac{1}{l} \sum_{j \in N(i)} C_j \, x_j, \tag{4.1}$$

where $N(i)$ is the neighborhood lens positions centered around x_i of size l, and $C_j \, x_j$ is the fitted value of x_j with a polynomial of degree m. Later, the final value of x_i^* is rounded

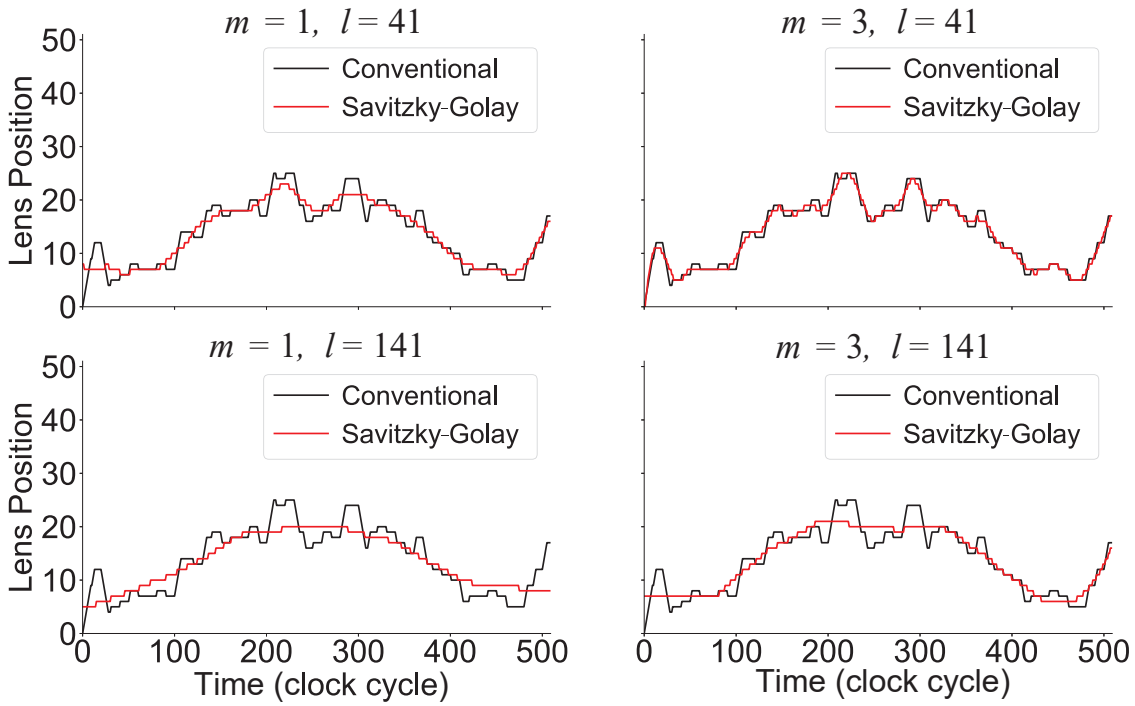


Figure 4.3: Savitzky-Golay method [1] applied on Scene 2 of the 9 FP objective. The term m is the polynomial degree and l is the time window size. By varying m and l, different smoothing trajectories can be applied—for example, using a lower m and larger l would result in stronger smoothing.

to the nearest integer to be a valid lens position.

There are two hyper-parameters m and l used to control the strength of the smoothness applied. Figure 4.3 demonstrates the effect of applying SG on Scene 2 of the 9 FP objective. As shown in this figure, different smoothing trajectories can be applied by varying m and l. For example, a lower polynomial degree m imposes less flexible smoothing compared to the higher one. Similarly, a larger window size l imposes stronger smoothing compared to the smaller one.

The MA and SG are *offline* smoothing methods because they apply a moving window centered around x_i. This cannot be achieved in a real camera system as it requires knowledge of future lens positions.

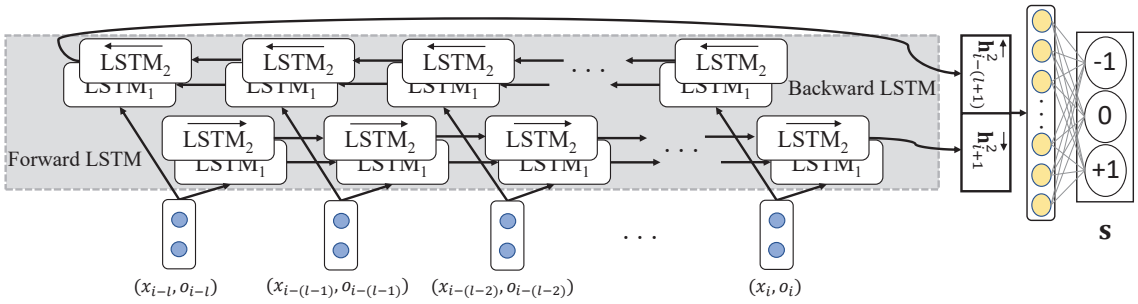


Figure 4.4: Our proposed bidirectional LSTM architecture of two LSTM layers followed by a fully connected layer. The term x_i is the current lens position and o_i is the optimal lens position (i.e., position where the image would be sharpest for a particular RoI) at clock cycle i. The output hidden states $\mathbf{h}$'s of both forward and backward LSTMs are concatenated to form our compact representation of time series data. The fully connected layer is added to output our score vector $\mathbf{s}$ that represents the scores of the three classes $\{-1, 0, 1\}$.

4.4 Supervised Online BLSTM

At clock cycle c, the BLSTM module receives the current lens position x_c and the associated optimal lens position o_c coming from PDAF module, as shown in Figure 4.2. BLSTM also observes the prior lens positions for a certain time window of size l from $c = i - l$ to $c = i$. The goal is to predict the smoothed lens position x^*_{i+1}. Based on the AF API implementation, a camera lens is allowed to move only one step per clock cycle, $|x_{i+1} - x_i| = \{0, 1\}, \forall i$. Therefore, the problem can be presented as a classification problem of three classes, $Y = \{-1, 0, 1\}$: move backward, no movement, move forward respectively. Thus, given the observed window of lens positions $\mathbf{X}_i = [(x_{i-l}, o_{i-l}), \ldots, (x_i, o_i)]$ and its associated label y_i, the task is to predict $f(\mathbf{X}_i)$ such that $x^*_{i+1} = x_i + f(\mathbf{X}_i)$.

Proposed BLSTM architecture Our model uses two LSTM layers followed by a fully connected layer that outputs the vector $\mathbf{s} \in \mathbb{R}^3$ as shown in Figure 4.4. The BLSTM architecture allows the output layer to capture information from past (backward) and future (forward) hidden states simultaneously. The vector $\mathbf{s}$ represents the scores of the three classes: $-1, 0, 1$. The vector $\mathbf{s}$ and forward/backward LSTMs are updated as follows:

$$\overrightarrow{\mathbf{h}}^1_{i+1} = \text{LSTM}_1(\overrightarrow{\mathbf{h}}^1_i, (x_i, o_i); \overrightarrow{\mathbf{W}}_1). \tag{4.2}$$

$$\overrightarrow{\mathbf{h}}^2_{i+1} = \text{LSTM}_2(\overrightarrow{\mathbf{h}}^2_i, \overrightarrow{\mathbf{h}}^1_{i+1}; \overrightarrow{\mathbf{W}}_2). \tag{4.3}$$

$$\overleftarrow{\mathbf{h}}^1_{i-1} = \text{LSTM}_1(\overleftarrow{\mathbf{h}}^1_i, (x_i, o_i); \overleftarrow{\mathbf{W}}_1). \tag{4.4}$$

$$\overleftarrow{\mathbf{h}}^2_{i-1} = \text{LSTM}_2(\overleftarrow{\mathbf{h}}^2_i, \overleftarrow{\mathbf{h}}^1_{i+1}; \overleftarrow{\mathbf{W}}_2). \tag{4.5}$$

$$\mathbf{s} = \mathbf{W}_{fc}[\overleftarrow{\mathbf{h}}^2_{i-(l+1)}, \overrightarrow{\mathbf{h}}^2_{i+1}] + \mathbf{b}_{fc}, \tag{4.6}$$

where LSTM_1 and LSTM_2 are the first and second LSTM layers respectively. The $\mathbf{W}$ terms denote the different weight matrices, $\mathbf{h}$'s are the hidden states of the LSTM units, and $\mathbf{b}_{fc}$ is the bias vector of the fully connected layer. Since our proposed architecture is a bidirectional LSTM, we use right and left arrows to indicate forward and backward passes.

We set the dimension for each LSTM unit to 32, which results in a 64-dimensional fully connected layer at the top. We fix the size of the input time window l to 20 as we found it effective empirically (see ablution study in Section 4.7). We use the Adam optimizer [90] to train our model with an initial learning rate of 0.0005, which is decreased by half every 1000 epochs. We train our model with mini-batches of size 128 using cross-entropy loss as follows:

$$\text{loss}(\mathbf{X}_i, y_i) = -\log\left(\frac{e^{\mathbf{s}[y_i]}}{\sum_j e^{\mathbf{s}[j]}}\right). \tag{4.7}$$

Cross-entropy loss is useful for classification problems where it increases as the predicted probability diverges from the actual label. During the training phase, we set the dropout rate to 0.1 to avoid overfitting. All the models described in the subsequent sections are implemented using Python with PyTorch framework [91] and trained with a NVIDIA TITAN X GPU. In our experiments, it takes roughly 10 minutes to complete 1000 training epochs. We set the maximum number of training epochs to 5000.

4.5 Unsupervised Online WMA

The MA method was introduced previously to generate the data required to train the supervised BLSTM. The MA method is offline and requires only input lens positions over time for smoothing and does not utilize the target optimal lens position that comes for free from the PDAF module. In this section, therefore, we introduce our online weighted moving average (WMA) method that uses the target optimal lens position to assign a

weight w_i to each lens position x_i as follows:

$$x_i^* = \frac{\sum_{j=i-l}^{i} w_j \, x_j}{\sum_{j=i-l}^{i} w_j}, \tag{4.8}$$

$$w_j = e^{-\beta(|x_j - o_j|)}, \tag{4.9}$$

where β is our smoothing rate, and o_j is the target optimal lens position. The final value of x_i^* is rounded to the nearest integer to be a valid lens position. The reason behind introducing the weight w_i is to encourage lens movement with a large difference from o_j and suppress lens movements in case of a small difference from the current position. Parameters β and l are the hyper-parameters used to control smoothing strength.

Unlike the *offline* smoothing methods (i.e., SG and MA), our *online* WMA uses only the prior data for x_i^* calculations in which it is considered as an *online* smoothing method. The *offline* smoothing methods cannot be performed in a real AF system because they require knowledge of where the lens will move in unseen frames.

4.6 Experiments and Discussions

4.6.1 Evaluation Criteria

Besides our qualitative comparisons, we also introduce four evaluation criteria for quantitative evaluations:

- Accuracy: calculates the number of correctly classified samples over the total number of samples. This accuracy is measured according to how similar our BLSTM predictions are to the *offline* smoothed lens motion.
- Lens motion reduction: reports the percentage of lens movements reduced after smoothing with respect to the corresponding conventional method.
- Sharpness change: calculates the percentage of sharpness change after smoothing with respect to the corresponding conventional method. The sharpness of RoI is calculated following the procedure in [14].
- Time delay: measures the time delay between the original trajectory and our *online* smoothed trajectory by applying the cross-correlation method in [92].

4.6.2 BLSTM Performance

The main purpose of this contribution is to validate the idea that the AF videos with smoothed lens motion are more preferred by users. We do not investigate the problem of which smoothing method is the best for such AF application. Furthermore, our results show that different smoothing methods have on par performance. In this section, therefore, we introduce our *online* smoothing method—i.e., BLSTM. The idea behind introducing our BLSTM model is to learn how to predict next lens movement in an *online* manner from the smoothed lens trajectories produced by *offline* smoothing methods (i.e., MA and SG).

Data preparation We first apply both *offline* MA and SG smoothing for each of the 36 videos. For the MA method, we set the window size $l = 41$ to apply a reasonable smoothing. For the SG method, we set window size $l = 141$ and polynomial degree $m = 3$ in order to impose strong smoothing with some flexibility. Next, we take the output smoothed lens positions to prepare the data for the BLSTM model as described in Section 4.4. We slide a time window of size 20 and stride of 1 to form our BLSTM input samples $\mathbf{X}_i$ and their corresponding y_i for each video data. We finally obtain two types of BLSTM models: (1) BLSTM-MA trained using the data produced by MA and (2) BLSTM-SG trained using the data produced by SG.

Training procedure We partition the data based on independent scenes into training, validation, and testing data. As described in Section 4.2, for each of the ten scenes, there are three fixed RoI objectives (global, 9 FP, and 51 FP) where sharpness is computed. For those objectives we divide the scenes into pairs $\{\{1, 8\}, \{2, 7\}, \{3, 6\}, \{4, 10\}, \{5, 9\}\}$, and then we perform cross-validation, where we take out-of-sample a pair of scenes one for validation and another for testing. Each objective, thus, has five pairs and we need to train five models to report results for the ten scenes. For the last objective FR (face region), only six scenes contain faces and for those we perform cross-validation by taking out-of-sample one scene for validation and testing. For the FR objective, we need to train six models to report results for the six scenes. The overall number of trained models is $3 \times 5 + 1 \times 6 = 21$ for each BLSTM type (i.e., BLSTM-MA and BLSTM-SG). We noticed that during training for some pairs there are lens positions that do not exist in the training set. For the network it is hard to predict for those values that are beyond the range of lens

position values in the training set. To tackle this issue, we generate a video of reasonable lens positions that covers all the possible values $0 - 50$ and augment this video in the training set.

Table 4.1: BLSTM-MA and BLSTM-SG accuracy of each objective for individual scenes. Overall, scene accuracy for all is mostly high, especially for the global objective, because it always has a single RoI and usually involves fewer lens movements. The 51 FP has slightly lower scene accuracy, because it has 51 RoIs and usually involves more lens movements. Recall that the FR objective can be applied on scenes that have faces and only six scenes contain faces.

Objective	**Scene**	**Accuracy** BLSTM MA	SG	**Objective**	**Scene**	**Accuracy** BLSTM MA	SG
Global	1	0.908	0.892	**9 FP**	1	0.992	0.982
	2	0.951	0.961		2	0.816	0.827
	3	0.969	0.986		3	0.798	0.875
	4	0.986	0.994		4	0.975	0.996
	5	0.993	0.993		5	0.964	0.986
	6	0.991	0.988		6	0.948	0.954
	7	0.994	0.994		7	0.951	0.941
	8	0.993	0.996		8	0.941	0.879
	9	0.997	0.992		9	0.918	0.884
	10	0.996	0.989		10	0.922	0.899
FR	1	–	–	**51 FP**	1	0.753	0.716
	2	–	–		2	0.853	0.824
	3	0.991	0.992		3	0.926	0.924
	4	0.900	0.810		4	0.845	0.773
	5	–	–		5	0.759	0.778
	6	0.907	0.870		6	0.910	0.865
	7	0.974	0.984		7	0.977	0.959
	8	0.983	0.986		8	0.897	0.888
	9	–	–		9	0.816	0.816
	10	0.938	0.943		10	0.932	0.825

Quantitative results Table 4.1 shows BLSTM-MA and BLSTM-SG accuracy of each objective for individual scenes. Our method achieves an average accuracy of 92.2%, and is especially good for the global objective. This is because the global objective has a single

region of interest (RoI) and usually involves fewer lens movements. Compared to other objectives, the 51 FP has slightly lower scene accuracy due to the fact it has 51 RoIs and usually involves more lens movements. In general, these results show that our proposed BLSTM is able to learn smoothed lens motion patterns and generalize for different scenes by testing on independent scenes that have never been seen by the network during training.

Table 4.2: Lens motion reduction and its effect on sharpness after applying smoothing methods for each objective. Compared with conventional methods, our *online* BLSTM has reduced lens motion notably with a very small loss in sharpness.

Objective	**Lens Motion Reduction**			
	Offline		*Online*	
	MA	SG	BLSTM-MA	BLSTM-SG
Global	42.83%	**48.17%**	**43.13%**	33.38%
9 FP	45.93%	**52.32%**	24.52%	**63.61%**
51 FP	45.27%	**49.54%**	36.66%	**40.75%**
FR	**29.15%**	25.53%	**25.28%**	11.20%

Objective	**Sharpness Change**			
	Offline		*Online*	
	MA	SG	BLSTM-MA	BLSTM-SG
Global	-0.12%	**-0.07%**	**-0.29%**	-0.47%
9 FP	**-0.31%**	-0.46%	**-1.15%**	-2.25%
51 FP	**-0.71%**	-1.19%	**-4.04%**	-5.17%
FR	**-0.32%**	-0.42%	**-0.45%**	-1.55%

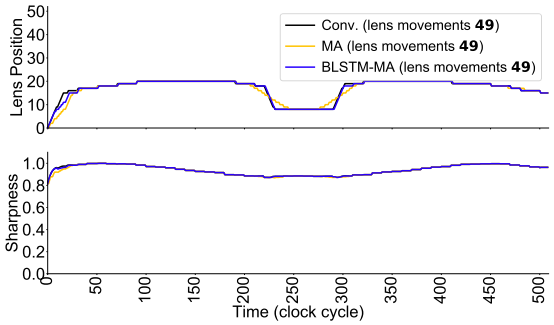
The accuracy in Table 4.1 evaluates the model based on the *offline* ground truth data. This accuracy metric suffers from cumulative error for *online* evaluations. For example, constructing lens movement trajectory for the whole video using *offline* evaluated samples results in propagating the error of any misclassified sample. To that end, we measure the trained BLSTM model's ability to predict *online* stream data of consecutive lens positions. This *online* BLSTM receives the optimal value o_c coming from the PDAF module at each clock cycle c where the lens position x_c^* is predicted based on the previously observed

sample $\mathbf{X}_{c-1}$. In a repeated way and by receiving o_c at a time, our *online* BLSTM is able to build the lens movements for the video.

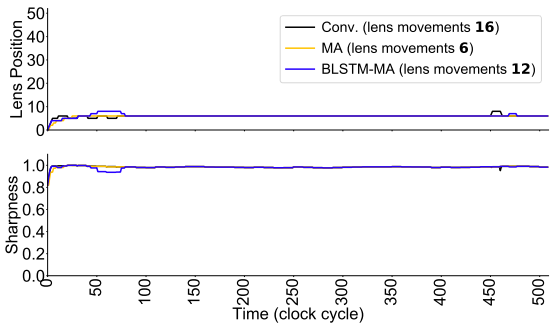
In Table 4.2, we report the lens motion reduction and sharpness change of smoothing methods applied for different AF objectives. Compared with the conventional AF, our *online* BLSTM is able to suppress small lens fluctuations without substantially affecting the sharpness value. These results show that our proposed *online* BLSTM is able to learn smoothed lens motion patterns from different *offline* methods (i.e., MA and SG) and performs almost as well as *offline* methods in an *online* manner.

Qualitative results Figure 4.5, 4.6, 4.7, and 4.8 illustrate the similarity between different lens motion trajectories: the conventional, the *online* BLSTM, and the *offline* one used for training. This figure also plots the corresponding sharpness values over time. Figure 4.5 and Figure 4.6 show a comparison between the conventional, BLSTM-MA, and MA methods applied on all the 36 videos as described in Section 4.2. Figure 4.7 and Figure 4.8 show a comparison between the conventional, BLSTM-SG, and SG methods applied on all the 36 videos as described in Section 4.2.

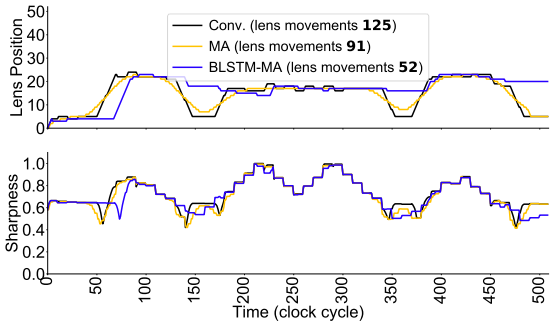
The total number of lens movements for each method is shown in the plot's legend. Our *online* BLSTM produces lens motion trajectories quite similar to the *offline* method trajectories used for training. Besides, our *online* BLSTM has notably fewer lens movements compared to the conventional one, in which fewer lens movements are highly preferred, as found in our user study (Section 3.4).



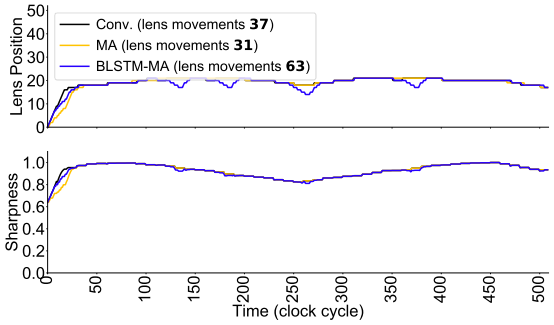
(a) Scene 1, global objective



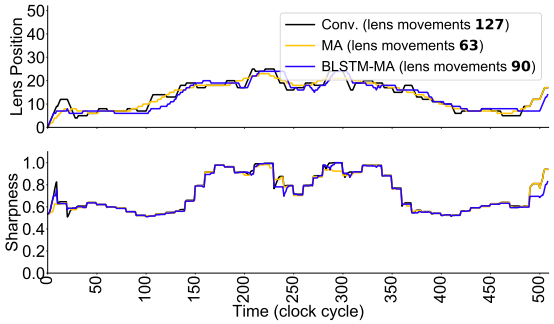
(b) Scene 1, 9-FP objective



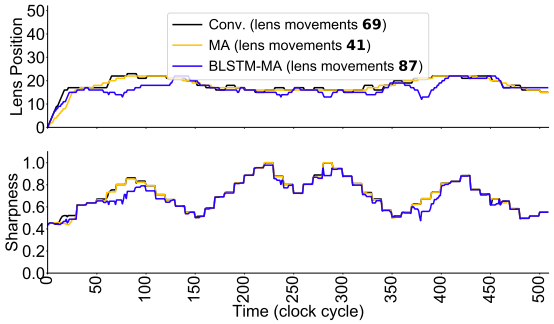
(c) Scene 1, 51-FP objective



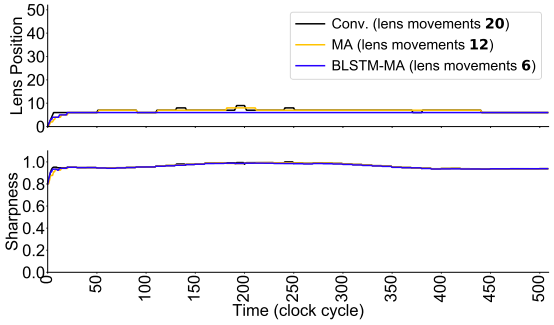
(d) Scene 2, global objective



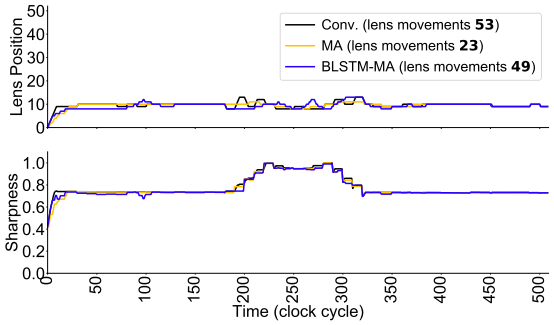
(e) Scene 2, 9-FP objective



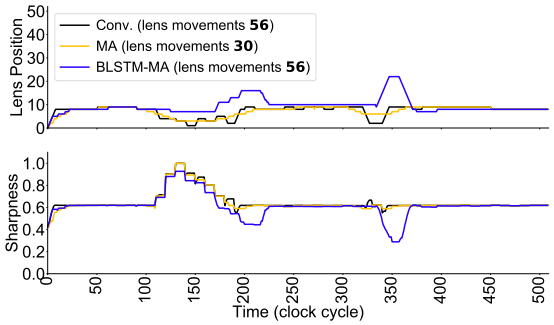
(f) Scene 2, 51-FP objective



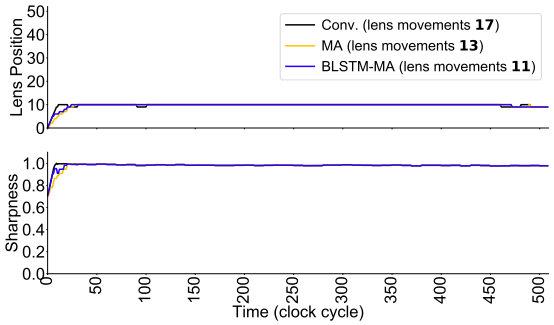
(g) Scene 3, global objective



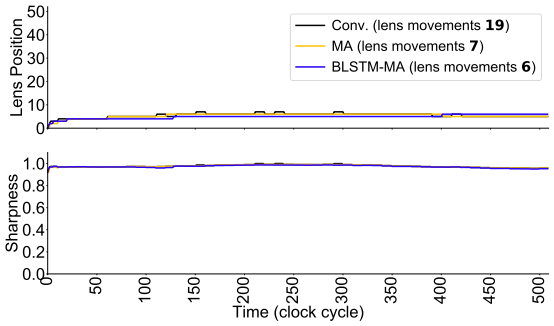
(h) Scene 3, 9-FP objective



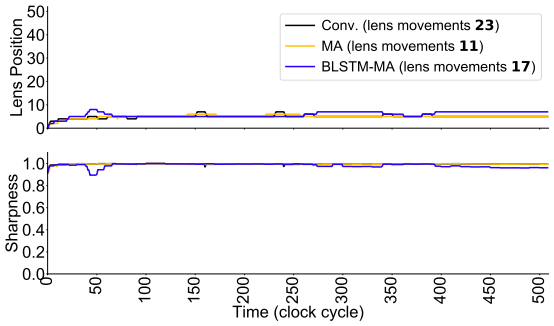
(i) Scene 3, 51-FP objective



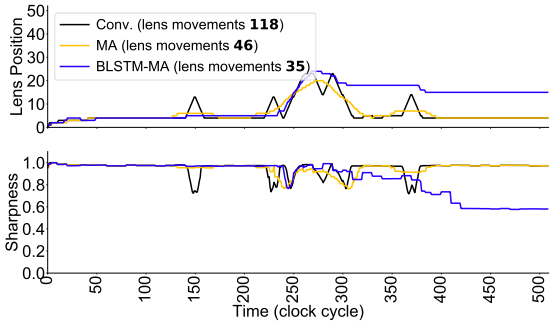
(j) Scene 3, FD objective



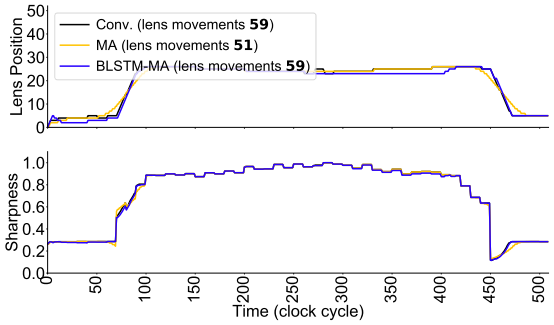
(k) Scene 4, global objective



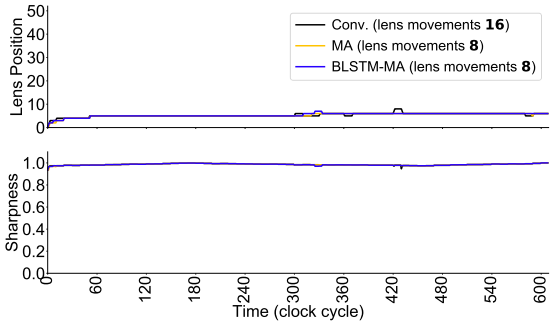
(l) Scene 4, 9-FP objective



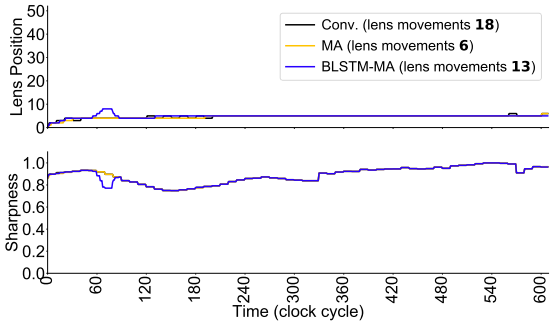
(m) Scene 4, 51-FP objective



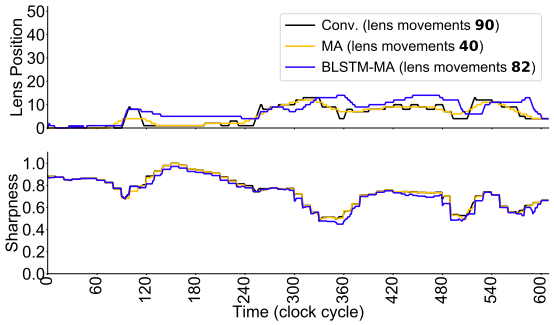
(n) Scene 4, FD objective



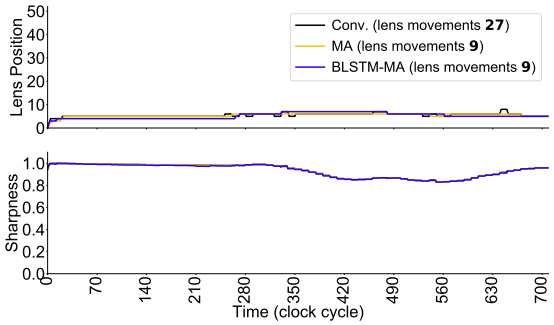
(o) Scene 5, global objective



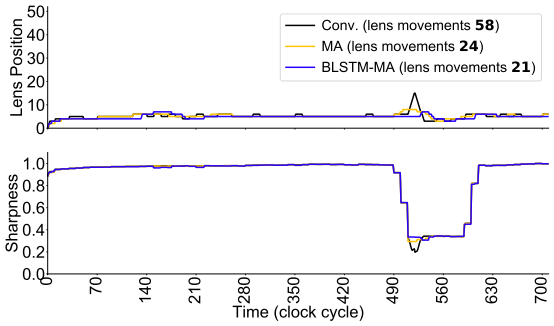
(p) Scene 5, 9-FP objective



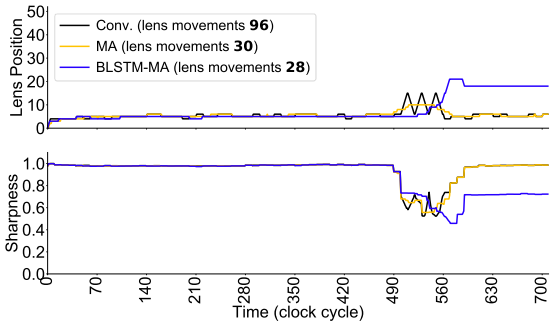
(q) Scene 5, 51-FP objective



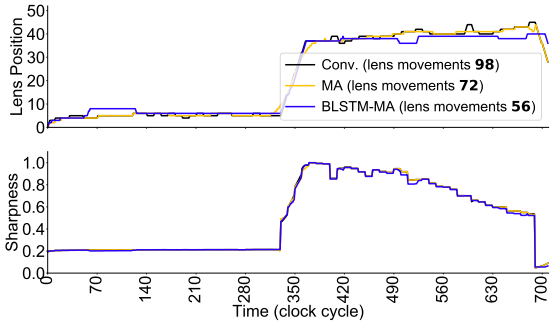
(r) Scene 6, global objective



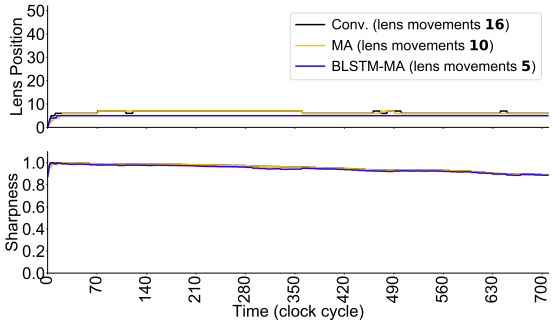
(s) Scene 6, 9-FP objective



(t) Scene 6, 51-FP objective

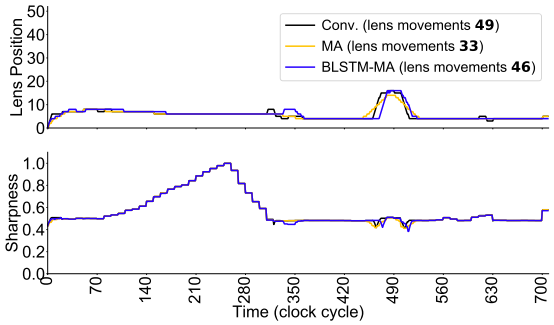


(u) Scene 6, FD objective

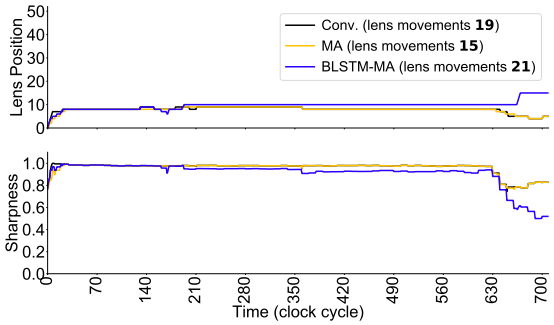


(v) Scene 7, global objective

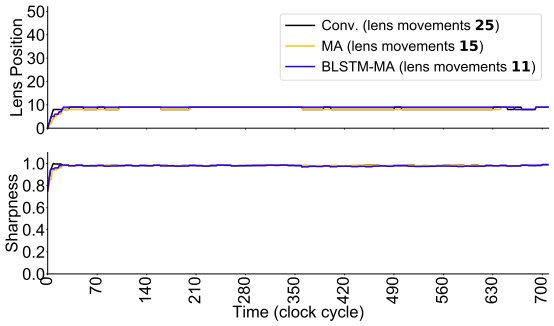
Figure 4.5: A comparison of lens positions between conventional AF and BLSTM-MA. *Offline* MA is also shown as this method is used to train the BLSTM-MA model. The plot above presents the lens positions over time and the one below shows the effect on sharpness on the same timeline. Total number of lens movements for each method is shown in the plot's legend. BLSTM-MA is able to suppress small lens fluctuations without substantially affecting the sharpness.



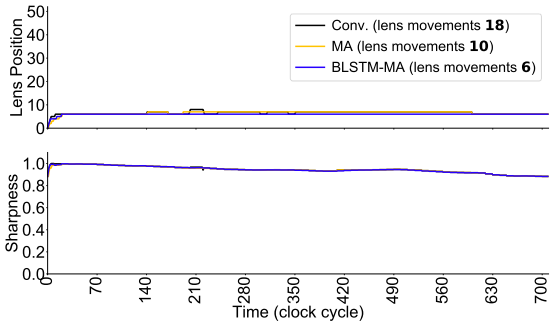
(a) Scene 7, 9-FP objective



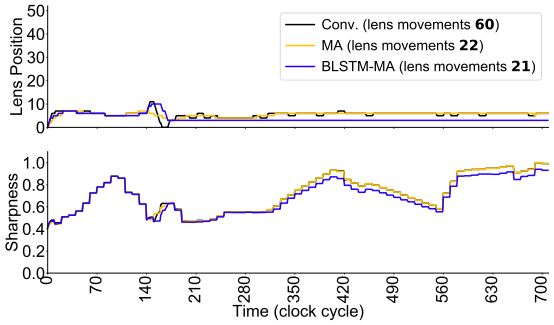
(b) Scene 7, 51-FP objective



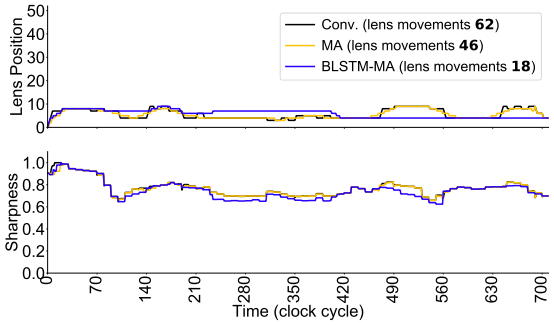
(c) Scene 7, FD objective



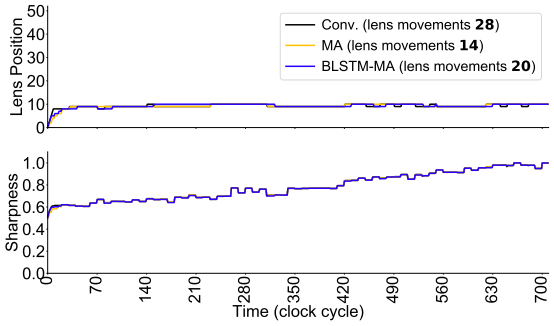
(d) Scene 8, global objective



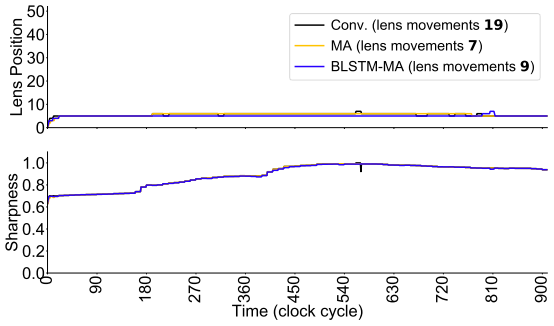
(e) Scene 8, 9-FP objective



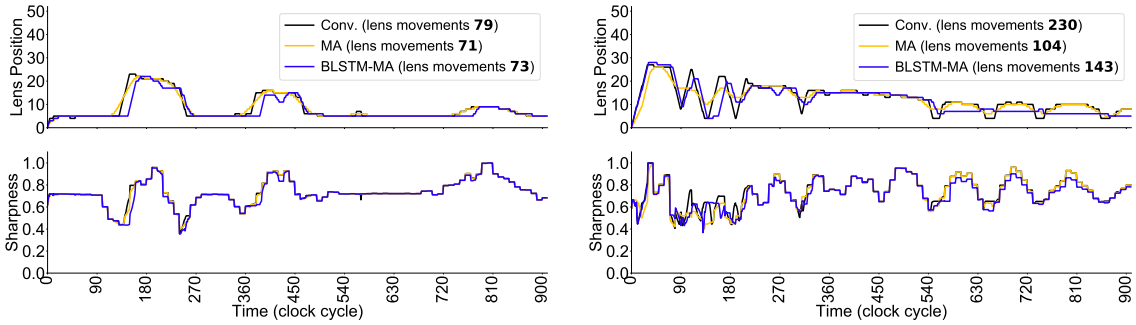
(f) Scene 8, 51-FP objective



(g) Scene 8, FD objective

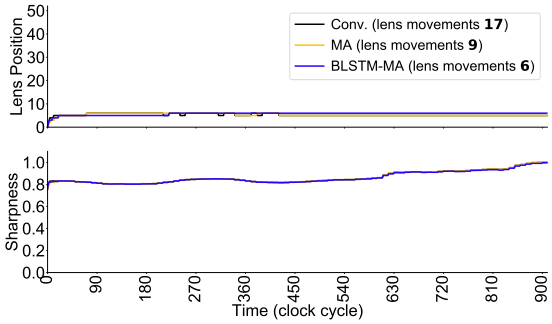


(h) Scene 9, global objective

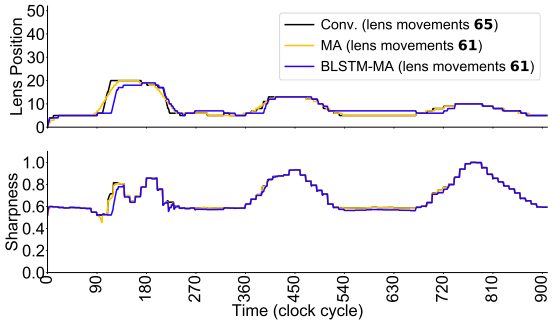


(i) Scene 9, 9-FP objective

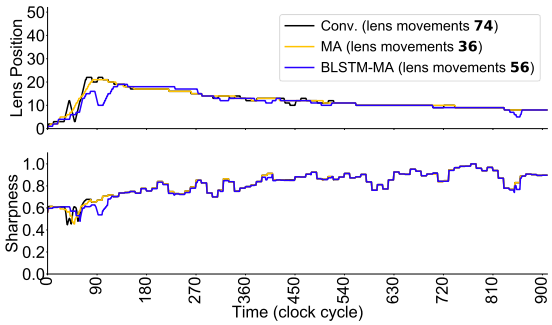
(j) Scene 9, 51-FP objective



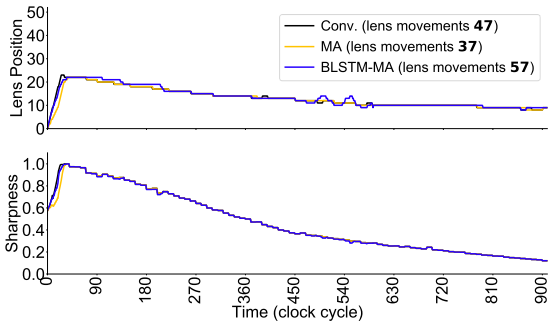
(k) Scene 10, global objective



(l) Scene 10, 9-FP objective

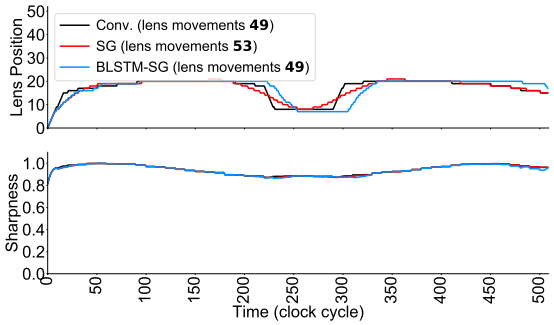


(m) Scene 10, 51-FP objective

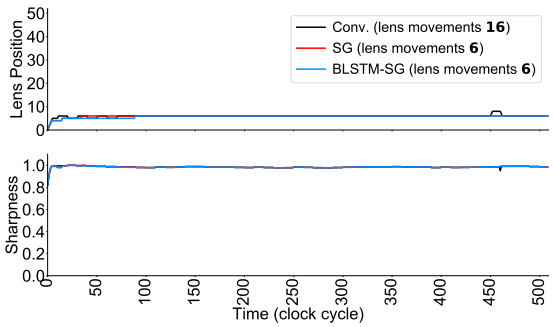


(n) Scene 10, FD objective

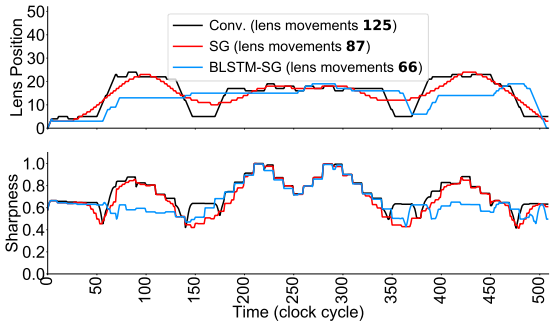
Figure 4.6: A comparison of lens positions between conventional AF and BLSTM-MA. *Offline* MA is also shown as this method is used to train the BLSTM-MA model. The plot above presents the lens positions over time and the one below shows the effect on sharpness on the same timeline. Total number of lens movements for each method is shown in the plot's legend. BLSTM-MA is able to suppress small lens fluctuations without substantially affecting the sharpness.



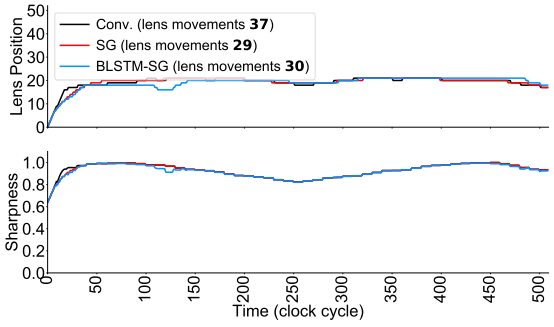
(a) Scene 1, global objective



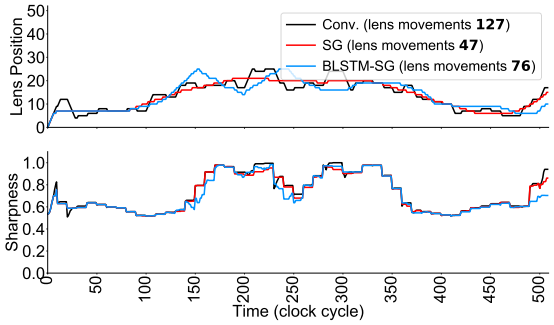
(b) Scene 1, 9-FP objective



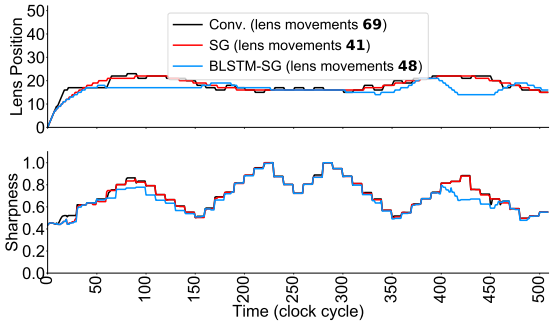
(c) Scene 1, 51-FP objective



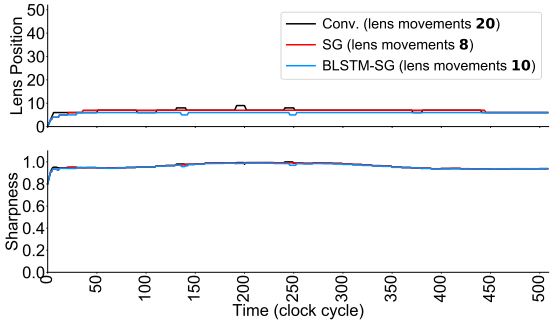
(d) Scene 2, global objective



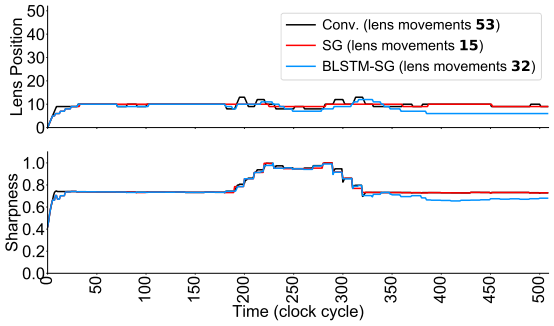
(e) Scene 2, 9-FP objective



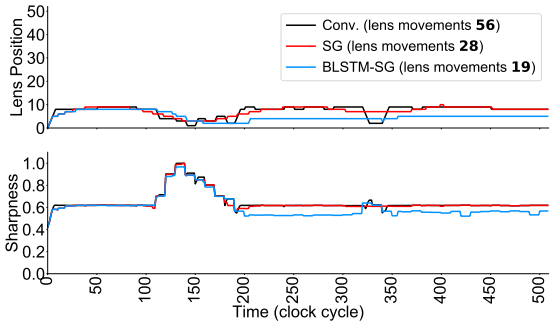
(f) Scene 2, 51-FP objective



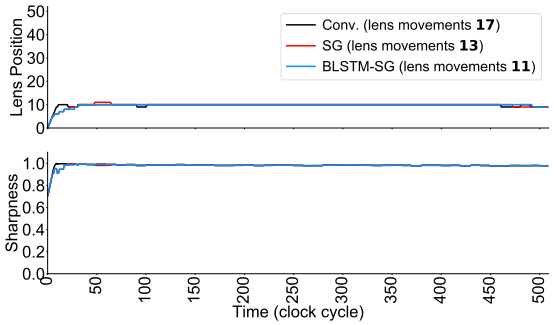
(g) Scene 3, global objective



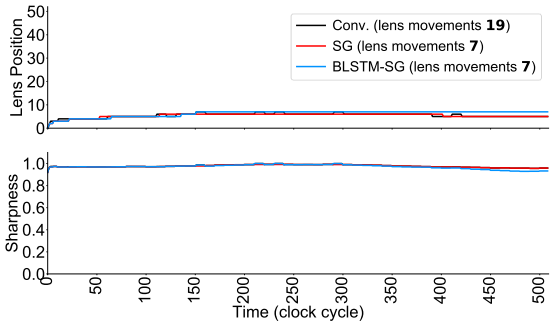
(h) Scene 3, 9-FP objective



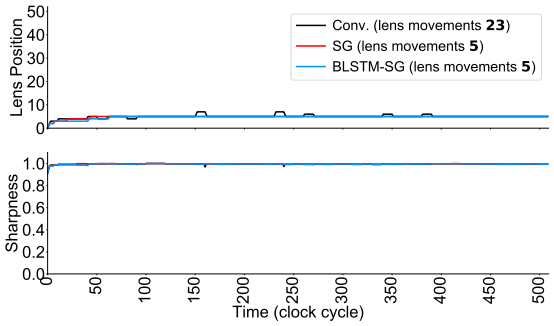
(i) Scene 3, 51-FP objective



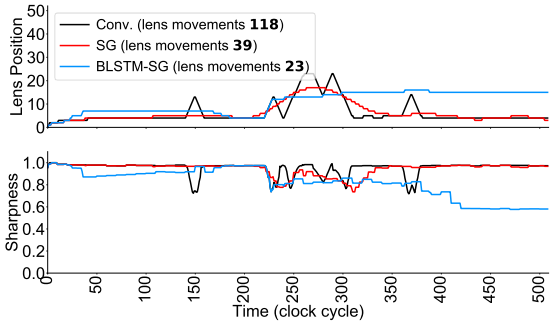
(j) Scene 3, FD objective



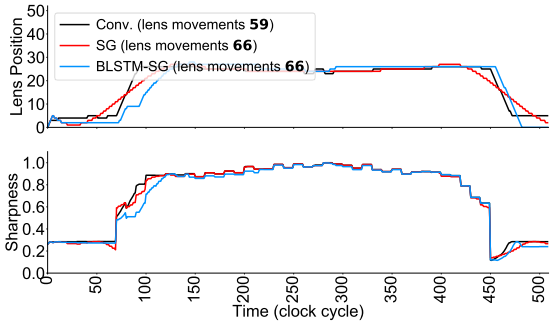
(k) Scene 4, global objective



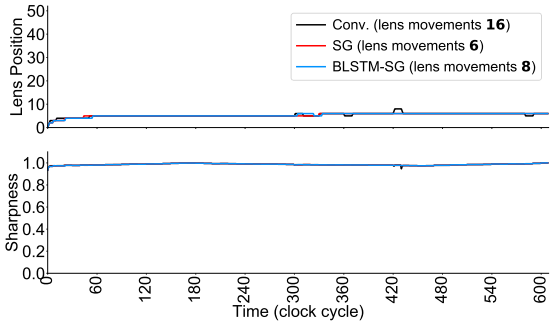
(l) Scene 4, 9-FP objective



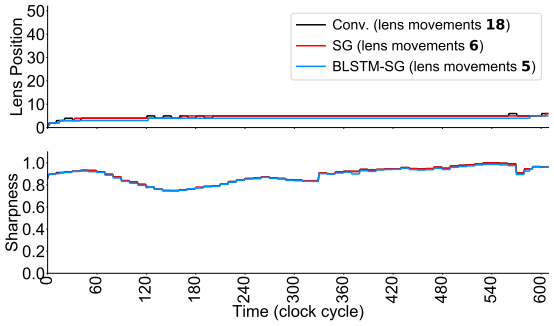
(m) Scene 4, 51-FP objective



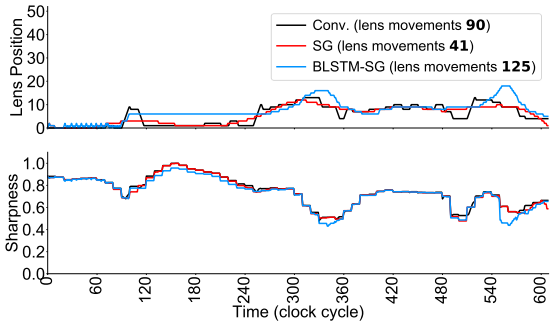
(n) Scene 4, FD objective



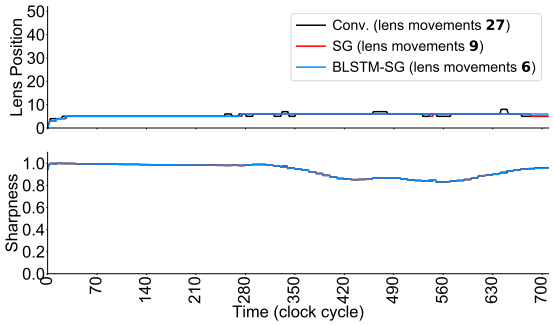
(o) Scene 5, global objective



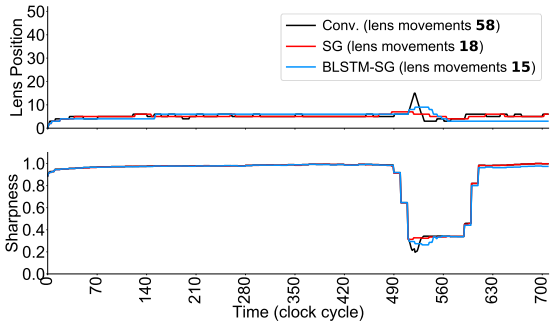
(p) Scene 5, 9-FP objective



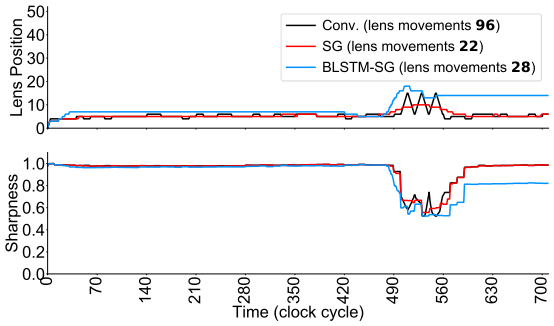
(q) Scene 5, 51-FP objective



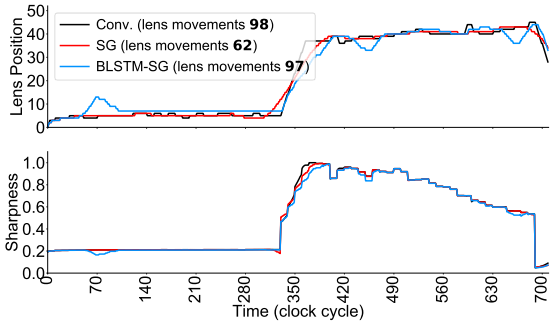
(r) Scene 6, global objective



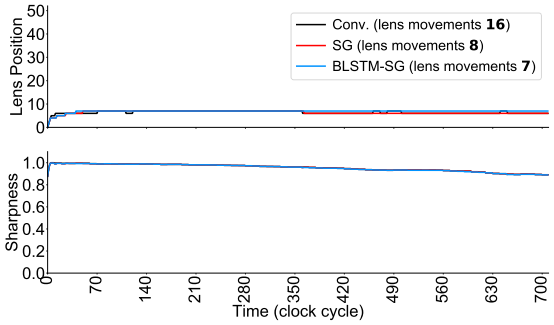
(s) Scene 6, 9-FP objective



(t) Scene 6, 51-FP objective

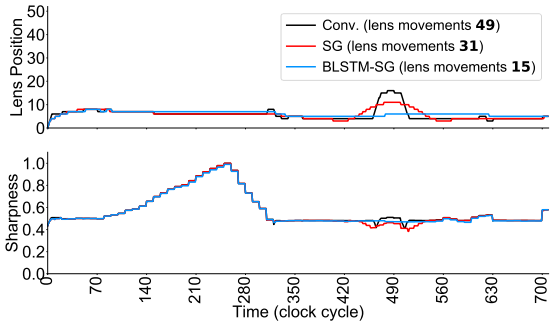


(u) Scene 6, FD objective

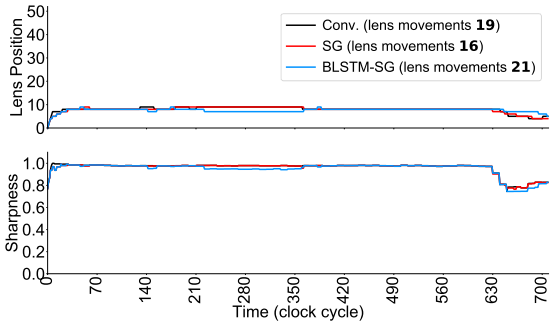


(v) Scene 7, global objective

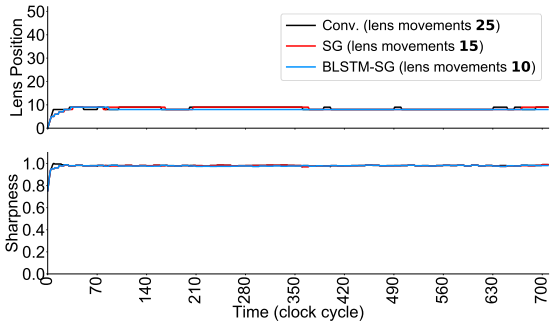
Figure 4.7: A comparison of lens positions between conventional AF and BLSTM-SG. *Offline* SG is also shown as this method is used to train the BLSTM-SG model. The plot above presents the lens positions over time and the one below shows the effect on sharpness on the same timeline. Total number of lens movements for each method is shown in the plot's legend. BLSTM-SG is able to suppress small lens fluctuations without substantially affecting the sharpness.



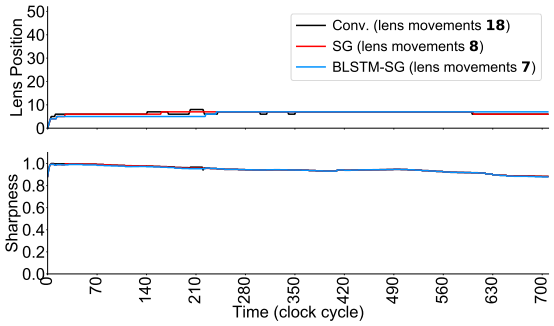
(a) Scene 7, 9-FP objective



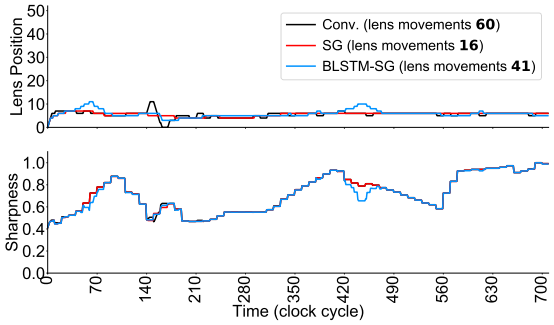
(b) Scene 7, 51-FP objective



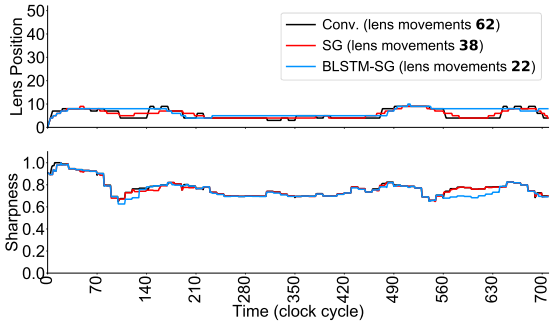
(c) Scene 7, FD objective



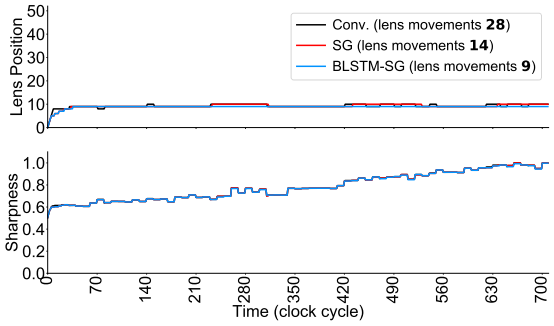
(d) Scene 8, global objective



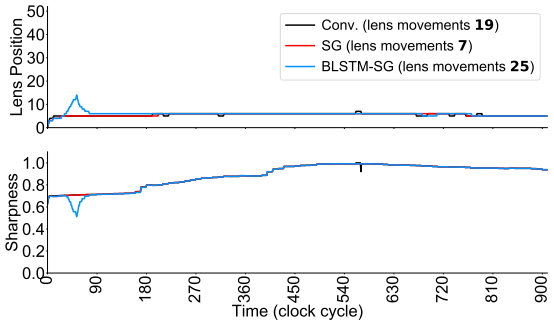
(e) Scene 8, 9-FP objective



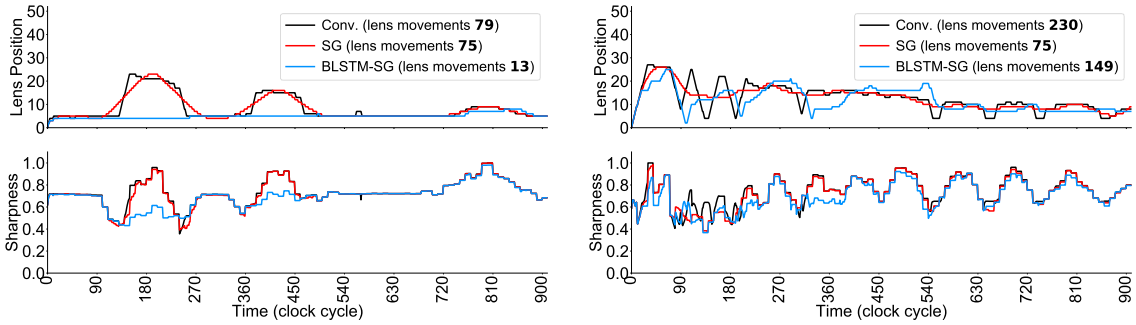
(f) Scene 8, 51-FP objective



(g) Scene 8, FD objective

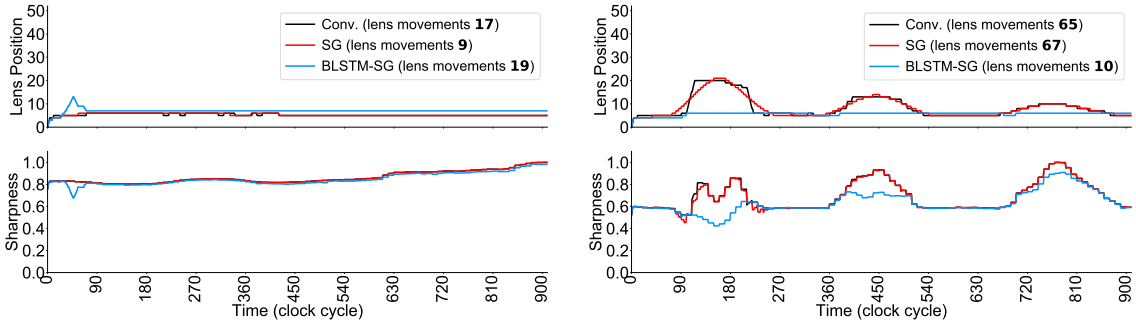


(h) Scene 9, global objective



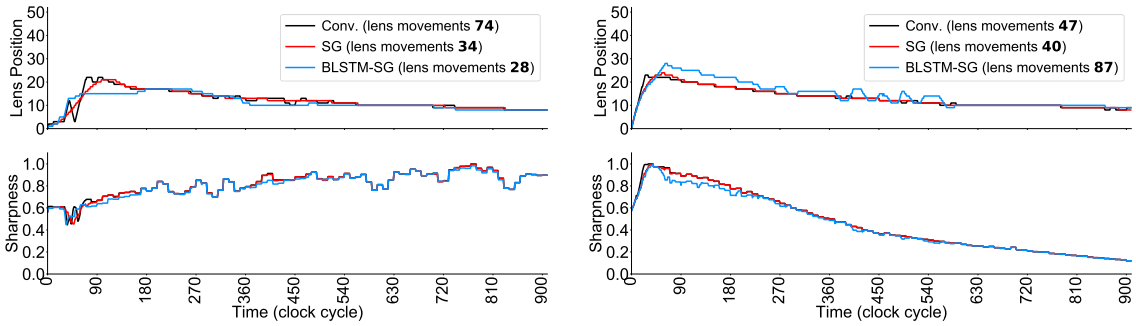
(i) Scene 9, 9-FP objective

(j) Scene 9, 51-FP objective



(k) Scene 10, global objective

(l) Scene 10, 9-FP objective



(m) Scene 10, 51-FP objective

(n) Scene 10, FD objective

Figure 4.8: A comparison of lens positions between the conventional AF and BLSTM-SG. *Offline* SG is also shown as this method is used to train the BLSTM-SG model. The plot above presents the lens positions over time and the one below shows the effect on sharpness on the same timeline. Total number of lens movements for each method is shown in the plot's legend. BLSTM-SG is able to suppress small lens fluctuations without substantially affecting the sharpness.

Objective	Accuracy	Objective	Accuracy
Global	0.979	**9-FP**	0.922
FR	0.931	**51-FP**	0.837

Table 4.3: Average accuracy for each objective. Overall, average accuracy for all is mostly high, especially for the global one, because it always has single RoI and usually involves fewer lens movements.

4.6.3 WMA Performance

In this section we present the quantitative and qualitative results of our *online* smoothing methods: (1) a supervised BLSTM model trained on *offline* generated smooth data using SG method and (2) an unsupervised WMA method. For the WMA method, β and l are set to 0.4 and 20, respectively.

Quantitative results Table 4.3 shows BLSTM average accuracy for each objective. Our BLSTM achieves an average accuracy of 91.6%, and is especially good for the global objective. This is because the global objective has a single RoI and usually involves fewer lens movements. Compared to other objectives, the 51-FP has slightly lower scene accuracy due to the fact it has 51 RoIs and usually involves more lens movements. In general, these results show that our proposed BLSTM is able to learn smoothed lens motion patterns and generalize for different scenes by testing on independent scenes that have never been seen by the network during training. Recall that the WMA method is unsupervised and we cannot evaluate its accuracy, because there is no ground truth data.

The accuracy in Table 4.3 evaluates the model based on the *offline* ground truth data. This accuracy metric suffers from cumulative error for *online* evaluations. For example, constructing lens movement trajectory for the whole video using *offline* evaluated samples results in propagating the error of any misclassified sample. To that end, we measure the trained BLSTM model's ability to predict *online* stream data of consecutive lens positions. This *online* BLSTM receives the optimal value o_i coming from the PDAF module at each clock cycle i where the lens position x_i^* is predicted based on the previously observed sample $\mathbf{X}_{i-1}$. In a repeated way and by receiving o_i at a time, our *online* BLSTM is able to build the lens movements for the whole video.

Table 4.4: Lens motion reduction and its effect on sharpness after applying smoothing methods for each objective. Compared with conventional methods, our *online* BLSTM and WMA have reduced lens motion notably, with a very small loss in sharpness. Bold numbers indicate the best results of comparing *online* methods only.

Objective	**Lens Motion Reduction**		
	Offline	*Online*	
	SG	BLSTM	WMA
Global	48.17%	33.38%	**44.49 %**
9-FP	52.32%	**63.61%**	50.17%
51-FP	49.54%	40.75%	**46.03**%
FR	25.53%	11.20%	**35.03%**

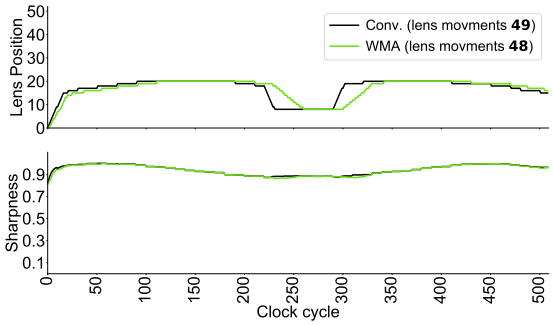
Objective	**Sharpness Change**		
	Offline	*Online*	
	SG	BLSTM	WMA
Global	-0.07%	-0.47%	**-0.13%**
9-FP	-0.46%	-2.25%	**-0.85%**
51-FP	-1.19%	-5.17%	**-0.91%**
FR	-0.42%	-1.55%	**-1.08%**

In Table 4.4, we report the lens motion reduction and sharpness change of smoothing methods applied for different AF objectives. Compared with the conventional AF, our *online* BLSTM and WMA are able to suppress the small lens fluctuations of the conventional lens positions while maintaining reasonable sharpness values. Interestingly, by looking at the overall performance, the unsupervised WMA performs slightly better than the supervised BLTM in terms of sharpness change. However, the WMA has a higher time delay compared to the BLSTM, where the WMA method achieved an average correlation of 0.88 and the BLSTM achieved a high average correlation of 0.96.

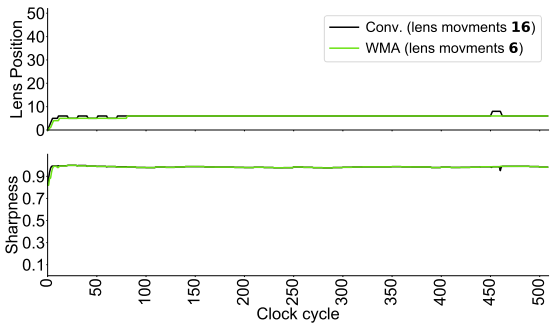
Qualitative results Figure 4.9 and Figure 4.10 illustrate the similarity between different lens motion trajectories for the *online* WMA. These figures also provide the corre-

sponding sharpness values over time. The total number of lens movements for each method is shown in the plot's legend. Our WMA have notably fewer lens movements compared to the conventional method, in which fewer lens movements are highly preferred. See project page[3] for additional examples, including videos and other visual comparisons.

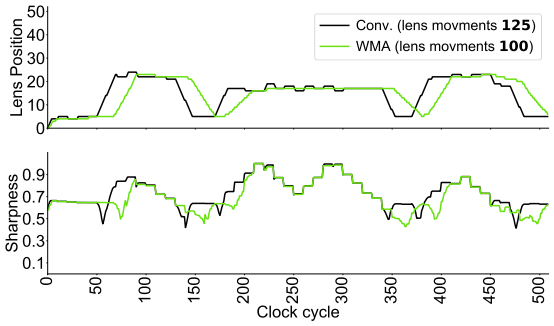
[3] https://abuolaim.nowaty.com/wacv_2020_autofocus_lens_motion/supplemental_materials/supplemental_materials.html



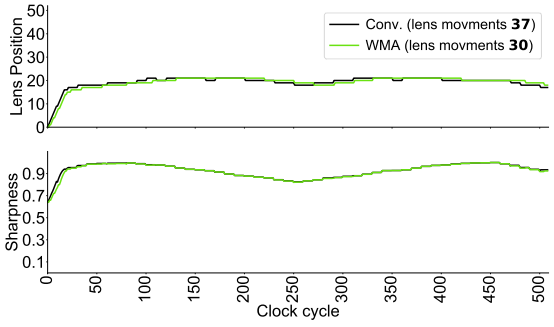
(a) Scene 1, global objective



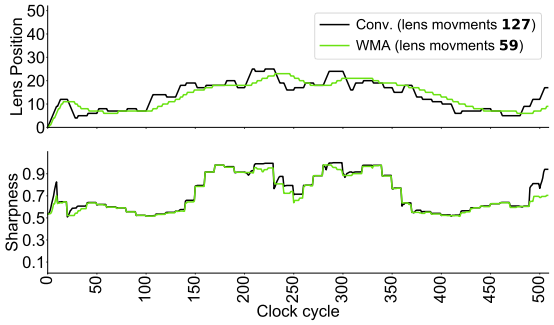
(b) Scene 1, 9-FP objective



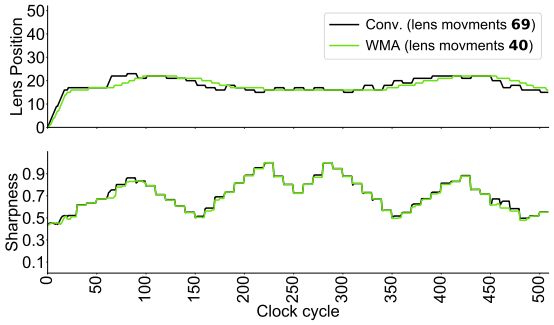
(c) Scene 1, 51-FP objective



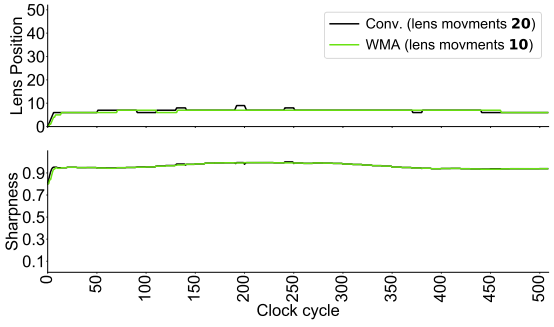
(d) Scene 2, global objective



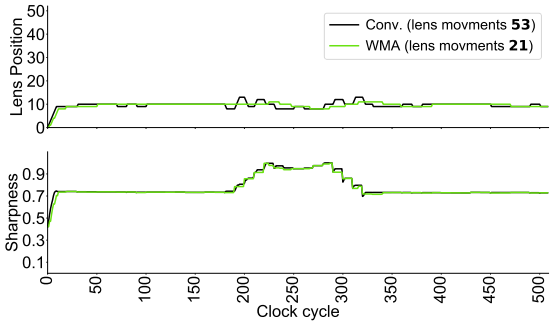
(e) Scene 2, 9-FP objective



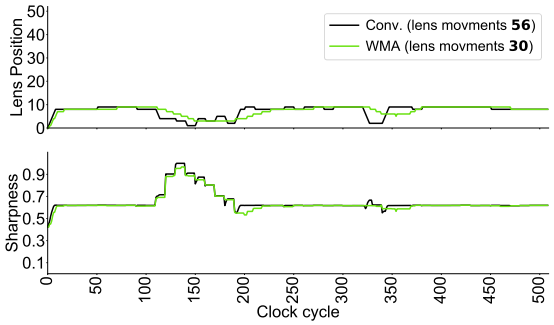
(f) Scene 2, 51-FP objective



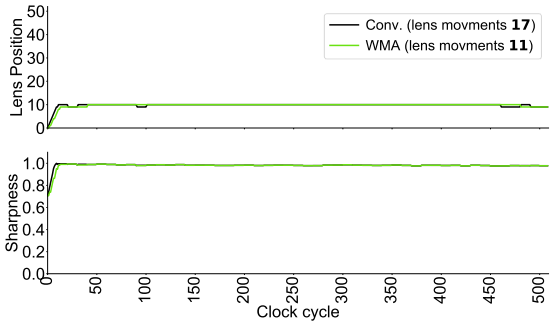
(g) Scene 3, global objective



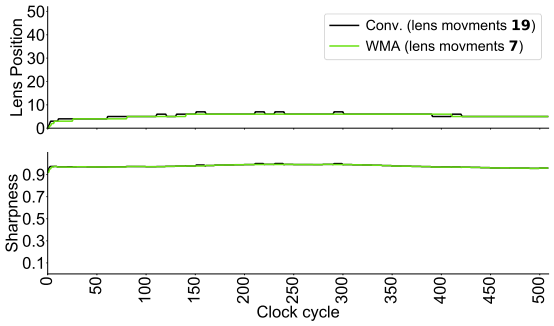
(h) Scene 3, 9-FP objective



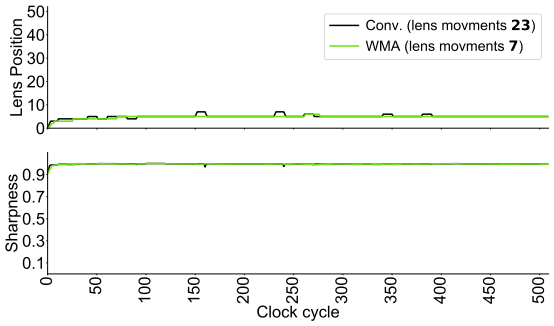
(i) Scene 3, 51-FP objective



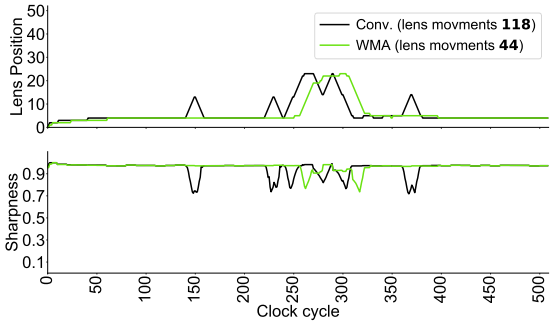
(j) Scene 3, FD objective



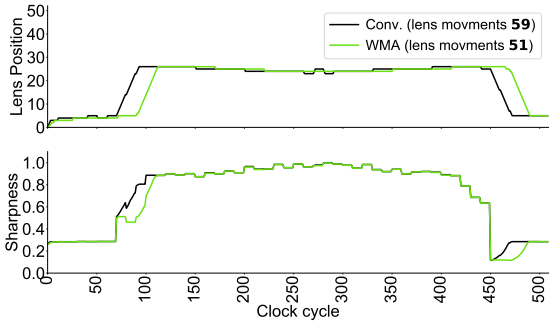
(k) Scene 4, global objective



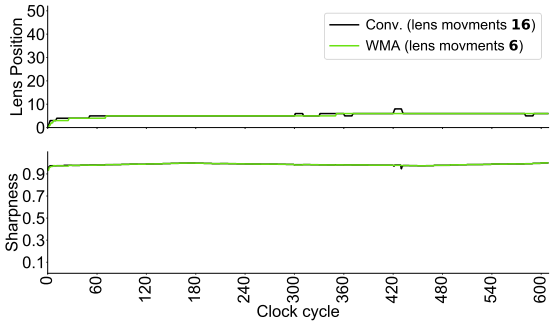
(l) Scene 4, 9-FP objective



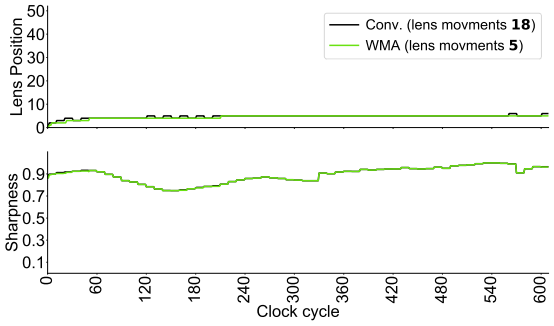
(m) Scene 4, 51-FP objective



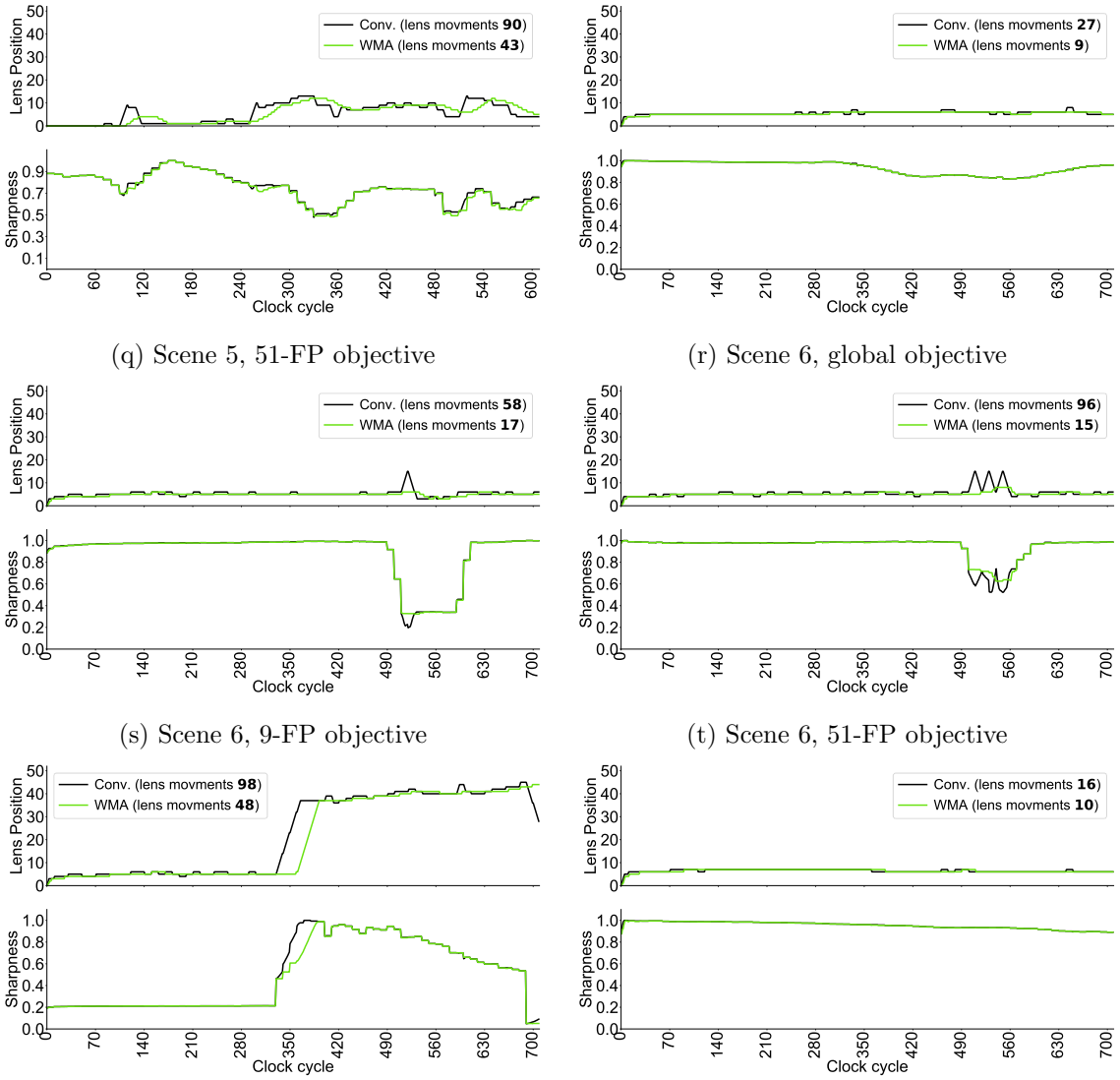
(n) Scene 4, FD objective



(o) Scene 5, global objective



(p) Scene 5, 9-FP objective



(q) Scene 5, 51-FP objective

(r) Scene 6, global objective

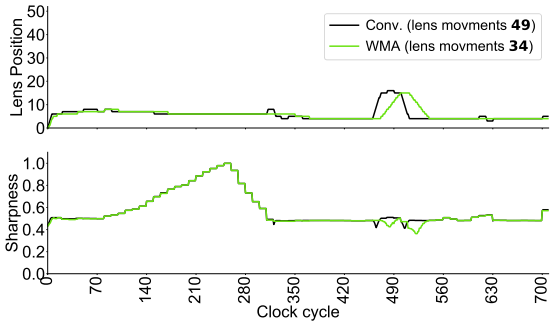
(s) Scene 6, 9-FP objective

(t) Scene 6, 51-FP objective

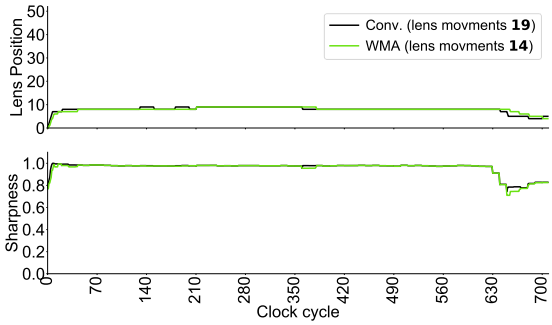
(u) Scene 6, FD objective

(v) Scene 7, global objective

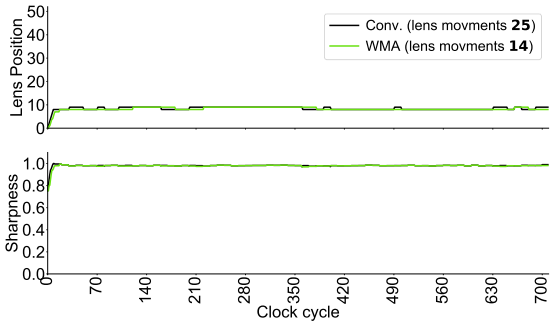
Figure 4.9: A comparison of lens positions between the conventional and WMA. The plot above presents the lens positions over time and the one below shows the effect on sharpness value on the same timeline. Total number of lens movements for each method is shown in the plot's legend between parentheses. WMA is able to suppress small lens fluctuations without substantially affecting the sharpness.



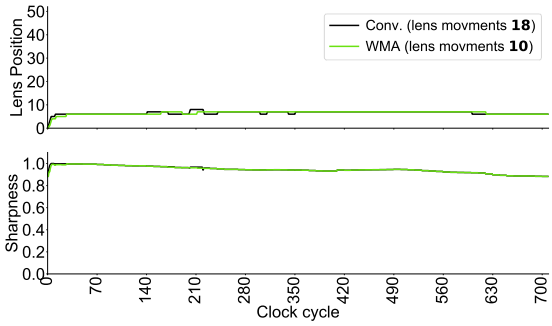
(a) Scene 7, 9-FP objective



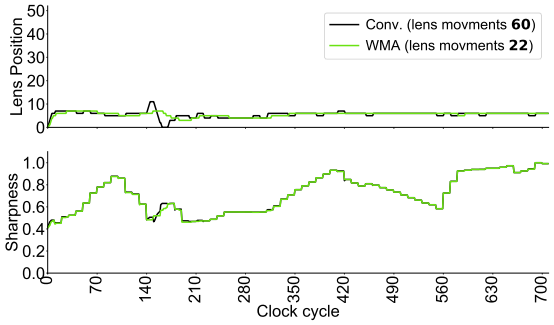
(b) Scene 7, 51-FP objective



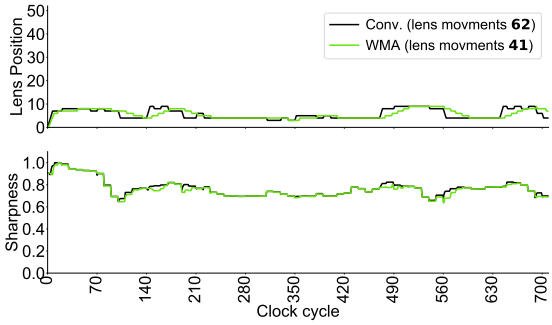
(c) Scene 7, FD objective



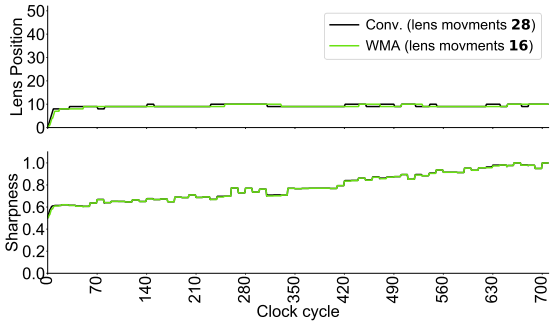
(d) Scene 8, global objective



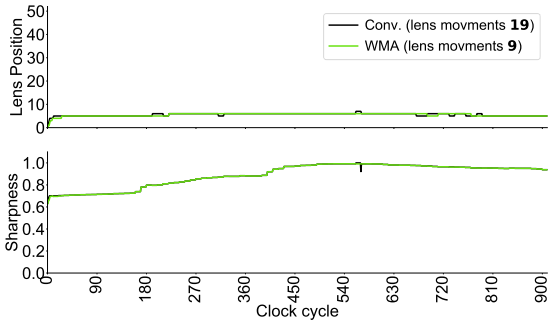
(e) Scene 8, 9-FP objective



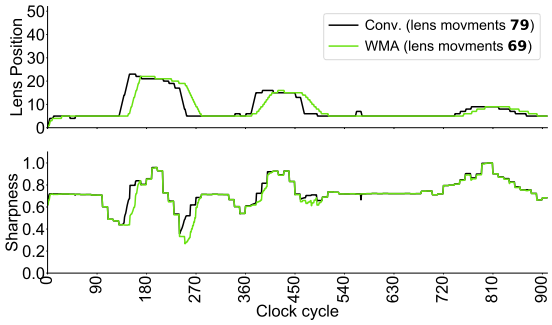
(f) Scene 8, 51-FP objective



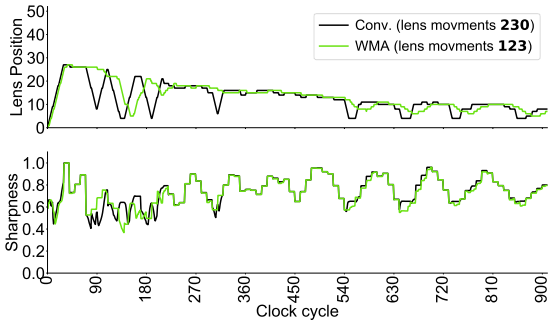
(g) Scene 8, FD objective



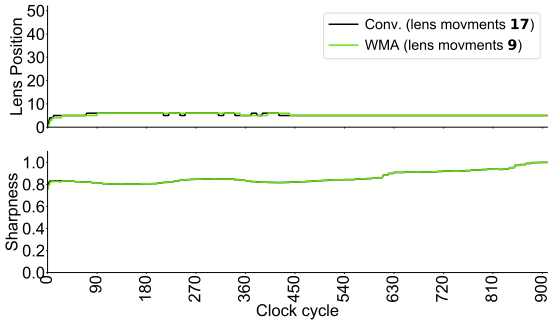
(h) Scene 9, global objective



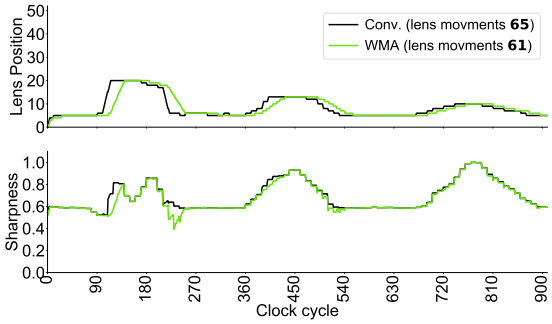
(i) Scene 9, 9-FP objective



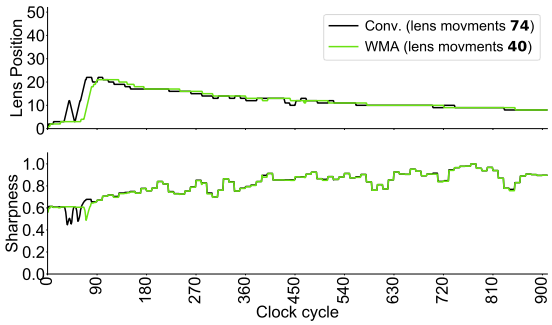
(j) Scene 9, 51-FP objective



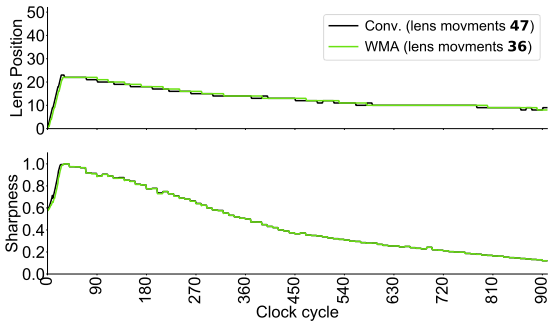
(k) Scene 10, global objective



(l) Scene 10, 9-FP objective



(m) Scene 10, 51-FP objective



(n) Scene 10, FD objective

Figure 4.10: A comparison of lens positions between the conventional and WMA. The plot above presents the lens positions over time and the one below shows the effect on sharpness value on the same timeline. Total number of lens movements for each method is shown in the plot's legend between parentheses. WMA is able to suppress small lens fluctuations without substantially affecting the sharpness.

4.7 Ablation Study

In this section we provide an ablation study of two variations of our BLSTM architecture: (1) a traditional LSTM architecture with one direction (forward pass only) and (2) our BLSTM with different input sizes.

Traditional LSTM vs. bidirectional LSTM Our first study, uses a traditional LSTM architecture with one direction (forward pass). This is referred to as a unidirectional LSTM (ULSTM). We use the same settings and hyperparameters used for our BLSTM, the only change is that we adopt a forward pass LSTM that results in a 32-dimensional fully connected layer at the top. This study is provided to investigate the effect of having a bidirectional LSTM and its ability to learn from two passes (i.e., forward and backward). In Table 4.5, we report the lens motion reduction and sharpness change of two *online* LSTM models (i.e., BLSTM and ULSTM) applied for different AF objectives. Overall, BLSTM has reduced lens motion more compared to ULSTM, except for BLSTM-MA on 9 FP and BLSTM-SG on FR. Moreover, BLSTM for all objectives has much smaller loss in sharpness compared to ULSTM. This Table shows that our BLSTM model has an advantage over the ULSTM, and learning from two passes has improved the LSTM's ability to learn smoothing patterns from *offline* smoothing methods with a slight loss in sharpness.

Table 4.5: A comparison between different LSTM architectures: bidirectional (BLSTM) vs. unidirectional (ULSTM). This table shows lens motion reduction and its effect on sharpness after applying different *online* LSTM smoothing methods. Compared to ULSTM, BLSTM has a better ability to learn smoothing patterns from *offline* smoothing methods with only a slight drop in sharpness.

Objective	**Lens Motion Reduction**			
	Trained Using MA Data		Trained Using SG Data	
	BLSTM	ULSTM	BLSTM	ULSTM
Global	**43.13%**	31.81%	**33.38%**	29.69%
9 FP	24.52%	**26.43%**	**63.61%**	43.83%
51 FP	**36.66%**	34.75%	**40.75%**	37.23%
FR	**25.28%**	20.85%	11.20%	**18.75%**

Objective	**Sharpness Change**			
	Trained Using MA Data		Trained Using SG Data	
	BLSTM	ULSTM	BLSTM	ULSTM
Global	**-0.29%**	-0.39%	**-0.47%**	-0.84%
9 FP	**-1.15%**	-1.97%	**-2.25%**	-2.42%
51 FP	**-4.04%**	-5.69%	**-5.17%**	-6.14%
FR	**-0.45%**	-1.17%	**-1.55%**	-7.55%

BLSTM with different input sizes Next, we examine a variant of our proposed method that uses the same architecture of our BLSTM, but adjusts the input size of time window l. We introduce our BLSTM with three different input sizes: BLSTM_{10} (l =10), BLSTM_{20} (l =20), and BLSTM_{40} (l =40). Table 4.6 presents the results of applying BLSTM with vary input size and show the amount of lens motion reduction with its effect on sharpness. By looking at the lens motion reduction table, BLSTM with a smaller window size imposes a larger reduction of lens motion in general. However, BLSTM with the smallest window size (i.e., BLSTM_{10}) also introduces more loss in sharpness compared to others. BLSTM_{20} imposes a reasonable reduction in lens motion and at the same time

has the smallest loss in sharpness for the most cases.

Table 4.6: A comparison of our BLSTM with three different input sizes: $BLSTM_{10}$ (l =10), $BLSTM_{20}$ (l =20), and $BLSTM_{40}$ (l =40). This table shows lens motion reduction and its effect on sharpness after applying different *online* BLSTM smoothing methods. $BLSTM_{20}$ imposes a reasonable reduction in lens motion and at the same time has the smallest loss in sharpness for the most cases.

Objective	**Lens Motion Reduction**					
	Trained Using MA Data			Trained Using SG Data		
	$BLSTM_{10}$	$BLSTM_{20}$	$BLSTM_{40}$	$BLSTM_{10}$	$BLSTM_{20}$	$BLSTM_{40}$
Global	**50.31%**	43.13%	28.90%	**43.45%**	33.38%	31.05%
9 FP	**44.02%**	24.52%	30.51%	**66.84%**	63.61%	16.47%
51 FP	**41.21%**	36.66%	29.04%	**58.96%**	40.75%	33.65%
FR	**31.63%**	25.28%	22.42%	06.33%	11.20%	**19.96%**

Objective	**Lens Motion Reduction**					
	Trained Using MA Data			Trained Using SG Data		
	$BLSTM_{10}$	$BLSTM_{20}$	$BLSTM_{40}$	$BLSTM_{10}$	$BLSTM_{20}$	$BLSTM_{40}$
Global	-0.43%	**-0.29%**	-0.30%	-1.06%	**-0.47%**	-0.61%
9 FP	-2.67%	-1.15%	**-0.58%**	-3.05%	**-2.25%**	-3.55%
51 FP	-4.52%	**-4.04%**	-4.25%	-8.76%	**-5.17%**	-5.66%
FR	-0.71%	**-0.45%**	-0.87%	-3.15%	-1.55%	**-1.00%**

4.8 User Study

A user study was conducted to investigate user preference for different smoothing methods against the conventional one. We conducted two user studies: (1) to determine if users prefer *offline* smoothed video over the conventional method, and (2) to determine if there is any particular preference for the different *online* smoothing methods.

For both studies, we defined scene number and objective as our independent variables, and user preference as our dependent variable. A force-choice paired comparison approach

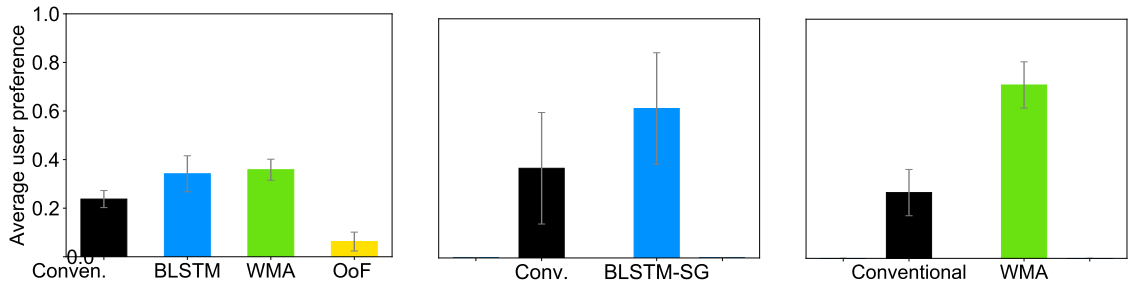


Figure 4.11: Results of our user study. In the first plot, all methods were preferred over the out-of-focus (OoF) method. Direct comparisons, in the second and third plots, show that BLSTM and WMA were significantly preferred over the conventional.

was adopted that requires each participant to select a single preferred video from a pair of videos. Both videos in a given pair are taken from the same scene and objective, but they have different methods.

For the first study, we picked six cases (scene with a certain objective) from the dataset (Section 4.2) to compare conventional vs. ideal. This study is provided to demonstrate the user preference for smoothed lens movements. For the second study, we compared user preference of the two *online* methods and the conventional method. We also introduced another dummy *online* method that is always out of focus with zero lens movements (maximum smoothness).

Specifically, five (5) scenes were used for the conventional, BLSTM, WMA, and out-of-focus (OoF) videos. The total number of paired comparisons for both studies is $6 \times \binom{2}{2} + 5 \times \binom{4}{2} = 36$.

For each video pair, eight opinions were collected from 32 participants (10 females and 22 males) ranging in age from 18 to 50. We designed a graphical user interface (GUI) that allows the user to view video pairs, one pair after the other. This interface allows the participant to watch the two videos in the current pair any number of times before making a selection of the preferred one and proceeding to the next pair. Each participant was shown nine pairs selected in a random manner. The survey takes on average four minutes to complete. The experiments were performed indoors with calibrated monitors and controlled lighting.

We aggregated user votes for each method into an overall score that represents user preference by counting the number of times each method is preferred over another. This

overall score is represented as a normalized average user preference. In addition, we calculated the 95% confidence intervals for these results and represented them by the error bars. For the first study, the right-hand plot in Figure 4.1 clearly shows the ideal smoothed methods were significantly preferred over the conventional one. The difference in the first study was statistically significant ($F = 49$, $p < 0.0002$). For the second study, the first plot in Figure 4.11 demonstrates the normalized average user preference for conventional vs. BLSTM vs. WMA vs. OoF. The difference in the second study was statistically significant ($F = 38.416$, $p < 0.000001$). The first plot in Figure 4.11 shows that all methods were preferred over OoF, even though it had zero lens movements, but it was the least preferred as it offered out-of-focus video. Both BLSTM and WMA were preferred over the conventional method as they smooth and reduce lens movements without affecting the video sharpness. WMA, with smaller error bar, was slightly preferred over BLSTM. The first plot does not reflect the actual number of times that BLSTM and WMA were selected over conventional, because we take the aggregate of user votes and the conventional method was selected many times over OoF. To show comparisons of two methods independently, we consider direct comparisons of conventional vs. BLSTM and conventional vs. WMA and present them in the second and third plots respectively. From the second and third plots we can clearly see that our proposed *online* methods (i.e., BLSTM and WMA) were significantly preferred.

4.9 Conclusion

This chapter has introduced two methods for video AF that perform *online* lens motion smoothing. Our method is motivated by the first user study in Section 3.4 that indicated users prefer smooth videos. While this may seem an intuitive finding, to the best of our knowledge, there is no literature explicitly examining smooth lens motion for video AF. This has motivated us to perform our own user study to show that users do prefer processed versions of conventional algorithms where the lens motion has been smoothed.

Based on this finding, we have proposed two algorithms to perform *online* lens smoothing. The first is a BLTSM learning-based method trained on smoothed lens trajectories. The second is a simple WMA. Both methods were significantly preferred over conventional AF. Interestingly, the simple WMA was slightly more preferred than the learning-based method. This is encouraging as the WMA can be easily incorporated into existing AF

hardware.

Our experiments were enabled by our recent AF dataset and research platform (Chapter 3). This dataset, however, contains only ten scenes with relatively low temporal sampling (yet was still over 33,000 images). This encourages the camera manufacturers to provide low-level hardware access to AF features to enable more research on this topic.

Part III

Post-Capture DoF Manipulation

Chapter 5

Extending Camera DoF

While Part II discussed the online lens adjustment in order to perform efficient image/video AF, this pre-capture adjustment does not always guarantee the desired DoF due to camera optics limitations such as:

- Wide aperture: results in blurred image details due to shallow DoF.
- Narrow aperture: large DoF may produce an all-in-focus image, making it hard to perform certain photography tricks like bokeh effect.

To this end, this part focuses on adjusting image DoF after effect in order to achieve the desired DoF in a post-processing step. This thesis part includes: image and video DoF extension given the DP sub-aperture views in Chapter 5 and Chapter 6, respectively. In Chapter 7, we address DoF extension from a single image without using DP data at inference time. To mimic the bokeh effect for blur-free images, we introduce our framework for synthesizing shallow DoF in Chapter 8.

5.1 Introduction

This chapter addresses the problem of reducing defocus blur (i.e., extending DoF). To understand why defocus blur is difficult to avoid, it is important to understand the mechanism governing image exposure. An image's exposure to light is controlled by adjusting two parameters: shutter speed and aperture size. The shutter speed controls the duration of light falling on the sensor, while the aperture controls the amount of light passing through the lens. The reciprocity between these two parameters allows the same exposure

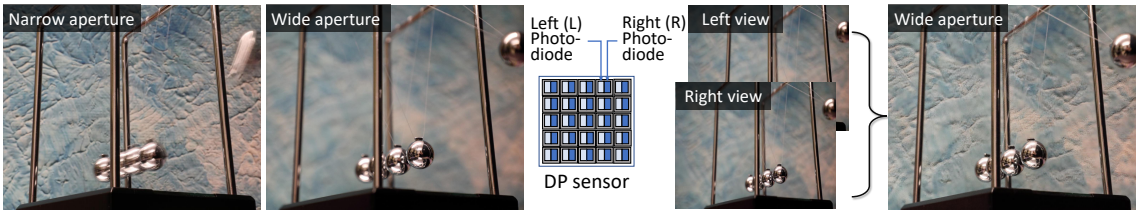


Figure 5.1: Images A and B are of the same scene and same approximate exposure. Image A is captured with a narrow aperture (f/22) and slow shutter speed. Image A has a wide depth of field (DoF) and little defocus blur, but exhibits motion blur from the moving object due to the long shutter speed. Image B is captured with a wide aperture (f/4) and a fast shutter speed. Image B exhibits defocus blur due to the shallow DoF, but has no motion blur. Our proposed DNN uses the two sub-aperture views from the dual-pixel sensor of image B to deblur image B, resulting in a much sharper image.

to occur by fixing one parameter and adjusting the other. For example, when a camera is placed in *aperture-priority* mode, the aperture remains fixed while the shutter speed is adjusted to control how long light is allowed to pass through the lens. The drawback is that a slow shutter speed can result in motion blur if the camera and/or an object in the scene moves while the shutter is open, as shown in Figure 5.1. Conversely, in *shutter-priority* mode, the shutter speed remains fixed while the aperture adjusts its size. The drawback of a variable aperture is that a wide aperture results in a shallow depth of field (DoF), causing defocus blur to occur in scene regions outside the DoF, as shown in Figure 5.1. There are many computer vision applications that require a wide aperture but still want an all-in-focus image. An excellent example is cameras on self-driving cars, or cameras on cars that map environments, where the camera must use a fixed shutter speed and the only way to get sufficient light is a wide aperture at the cost of defocus blur.

Our aim is to reduce the unwanted defocus blur. The novelty of our approach lies in the use of data available from dual-pixel (DP) sensors used by modern cameras. DP sensors are designed with two photodiodes at each pixel location on the sensor. The DP design provides the functionality of a simple two-sample light-field camera and was developed to improve how cameras perform autofocus. Specifically, the two-sample light-field provides two sub-aperture views of the scene, denoted in this chapter as *left* and *right* views. The

light rays coming from scene points that are within the camera's DoF (i.e., points that are in focus) will have no difference in phase between the left and right views. However, light rays coming from scene points outside the camera's DoF (i.e., points that are out of focus) will exhibit a detectable disparity in the left/right views that is directly correlated to the amount of defocus blur. We refer to it as *defocus disparity*. Cameras use this phase shift information to determine how to move the lens to focus on a particular location in the scene. After autofocus calculations are performed, the DP information is discarded by the camera's hardware.

Contribution We propose a DNN to perform defocus deblurring that uses the DP images from the sensor available at capture time. In order to train the proposed DNN, a new dataset of 500 carefully captured images exhibiting defocus blur and their corresponding all-in-focus image is collected. This dataset consists of 2000 images – 500 DoF blurred images with their 1000 DP sub-aperture views and 500 corresponding all-in-focus images – all at full-frame resolution (i.e., 6720×4480 pixels). Using this training data, we propose a DNN architecture that is trained in an end-to-end manner to directly estimate a sharp image from the left/right DP views of the defocused input image. Our approach is evaluated against conventional methods that use only a single input image and show that our approach outperforms the existing state-of-the-art approaches in both signal processing and perceptual metrics. Most importantly, the proposed method works by using the DP sensor images that are a free by-product of modern image capture systems. The contributions of this chapter were published in the European Conference on Computer Vision (ECCV), 2020 [3].

Related links:

- Project page:
  `https://abuolaim.nowaty.com/eccv_2020_dp_defocus_deblurring/`
- Dataset, code, and trained models:
  `https://github.com/Abdullah-Abuolaim/defocus-deblurring-dual-pixel`

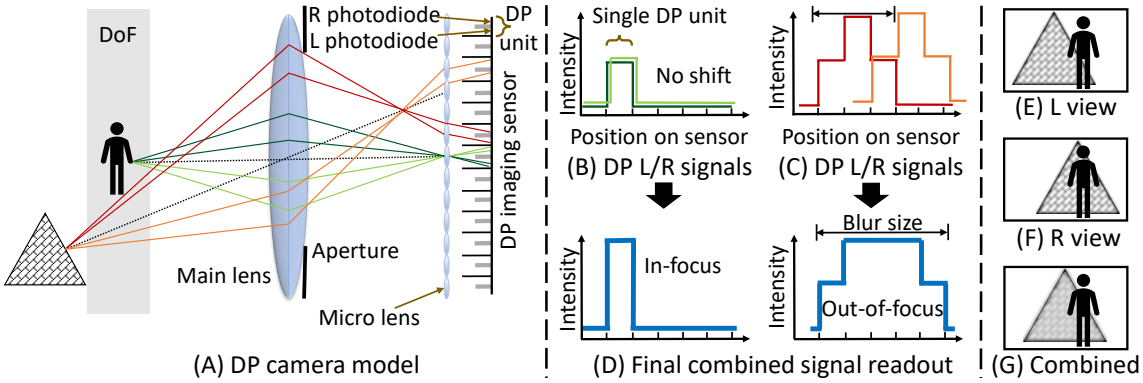


Figure 5.2: Image formation diagram for a DP sensor. (A) Shows a thin-lens camera and a DP sensor. The light rays from different halves of the main lens fall on different left and right photodiodes. (B) Scene points that are within the DoF (highlighted in gray) have no phase shift between their L/R views. Scene points outside DoF have a phase shift as shown in (C). The L/R signals are aggregated and the corresponding combined signal is shown in (D). The blur size of the L signal is smaller than the combined one in the out-of-focus case. The defocus disparity is noticeable between the captured L/R images (see (E) and (F)). The final combined image in (G) has more blur. Our DNN leverages this additional information available in the L/R views for image defocus deblurring.

5.2 DP Image Formation

We begin with a brief overview of the DP image formation. As previously mentioned, the DP sensor was designed to improve camera auto-focus technology. Figure 5.2 shows an illustrative example of how DP imaging works and how the left/right images are formed. A DP sensor provides a pair of photodiodes for each pixel with a microlens placed at the pixel site, as shown in Figure 5.2-A. This DP unit arrangement allows each pair of photodiodes (i.e., dual-pixel) to record the light rays independently. Depending on the sensor's orientation, this arrangement can be shown as left/right or top/down pair; in this chapter, we refer to them as the left/right pair – or L and R. The difference between the two views is related to the defocus amount at that scene point, where out-of-focus scene points will have a difference in phase and be blurred in opposite directions using a point spread function (PSF) and its flipped one [4]. This difference yields noticeable defocus disparity that is correlated to the amount of defocus blur.

The phase-shift process is illustrated in Figure 5.2. The person shown in Figure 5.2-

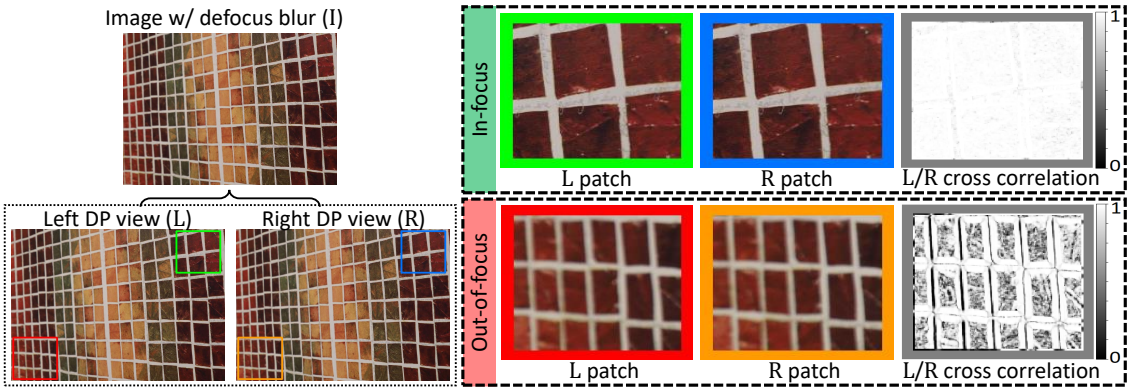


Figure 5.3: An input image I is shown with a spatially varying defocus blur. The two dual-pixel (DP) images (L and R) corresponding to I are captured at imaging time. In-focus and out-of-focus patches in the L and R DP image patches exhibit different amounts of pixel disparity as shown by the cross-correlation of the two patches. This information helps the DNN to learn the extent of blur in different regions of the image.

A is within the camera's DoF, as highlighted in gray, whereas the textured pyramid is outside the DoF. The light rays from the in-focus object converge at a single DP unit on the imaging sensor, resulting in an in-focus pixel and no disparity between their DP L/R views (Figure 5.2-B). The light rays coming from the out-of-focus regions spread across multiple DP units and therefore produce a difference between their DP L/R views, as shown in Figure 5.2-C. Intuitively, this information can be exploited by a DNN to learn where regions of the image exhibit blur and the extent of this blur. The final output image is a combination of the L/R views, as shown in Figure 5.2-G.

By examining real examples shown in Figure 5.3 it becomes apparent how a DNN can leverage these two sub-aperture views as input to deblur the image. In particular, patches containing regions that are out-of-focus will exhibit a notable defocus disparity in the two views that is directly correlated to the amount of defocus blur. By training a DNN with sufficient examples of the L/R views and the corresponding all-in-focus image, the DNN can learn how to detect and correct blurred regions. Animated examples of the difference between the DP views is provided in Figure 5.4.

Figure 5.4: Animated ground-truth DP views. **Note: The DP views are animated; click on the image to start the animation. It is recommended to open this PDF in Adobe Acrobat Reader to work properly.**

5.3 Dataset Collection

Our first task is to collect a dataset with the necessary DP information for training our DNN. While most consumer cameras employ PDAF sensors, we are aware of only two camera manufacturers that provide DP data – Google and Canon. Specifically, Google's research team has released an application to read DP data [93] from the Google Pixel 3 and 4 smartphones. However, smartphone cameras are currently not suitable for our problem for two reasons. First, smartphone cameras use fixed apertures that cannot be adjusted for data collection. Second, smartphone cameras have narrow aperture and exhibit large DoF; in fact, most cameras go to great lengths to simulate shallow DoF by purposely introducing defocus blur [27]. As a result, our dataset is captured using a Canon EOS 5D Mark IV DSLR camera, which provides the ability to save and extract full-frame DP

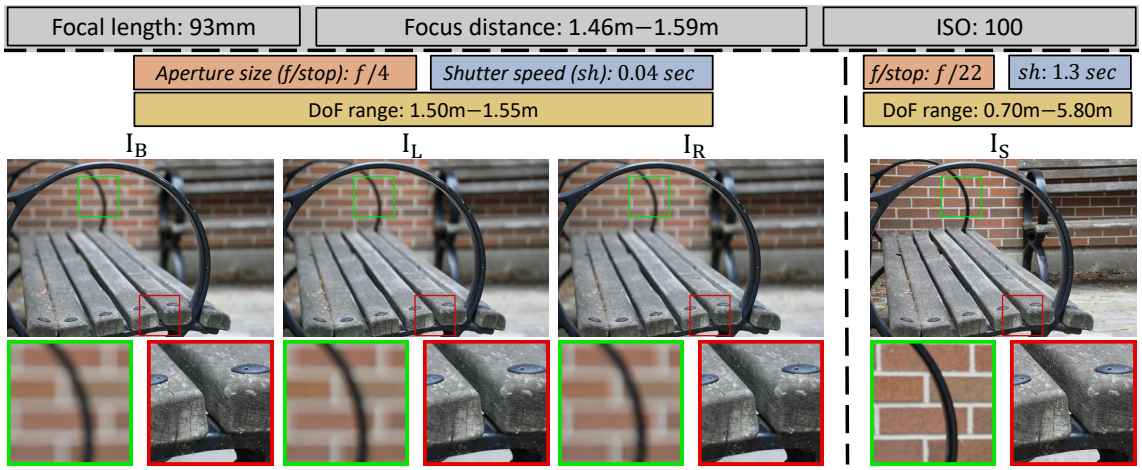


Figure 5.5: An example of an image pair with the camera settings used for capturing. $\mathrm{I_L}$ and $\mathrm{I_R}$ represent the Left and Right DP views extracted from $\mathrm{I_B}$. The focal length, ISO, and focus distance are fixed between the two captures of $\mathrm{I_B}$ and $\mathrm{I_S}$. The aperture size is different, and hence the shutter speed and DoF are accordingly different too. In-focus and out-of-focus zoomed-in patches are extracted from each image and shown in green and red boxes, respectively.

images.

Using the Canon camera, we capture a pair of images of the same static scene at two aperture sizes – $f/4$ and $f/22$ – which are the maximum (widest) and minimum (narrowest) apertures possible for our lens configuration. The lens position and focal length remain fixed during image capture. Scenes are captured in aperture-priority mode, in which the exposure compensation between the image pairs is done automatically by adjusting the shutter speed. The image captured at $f/4$ has the smallest DoF and results in the blurred input image $\mathrm{I_B}$. The image captured at $f/22$ has the largest DoF and serves as the all-in-focus target image denoted as $\mathrm{I_S}$ (sharp image). Focus distance and focal length differ across captured pairs in order to capture a diverse range of defocus blur types. Our captured images offer the following benefits over prior datasets:

- **High-quality images**. Our captured images are low-noise images (i.e., low ISO equates to low-noise [94]) and of full resolution of 6720×4480. All images, including the left/right DP views, are processed to an sRGB and encoded with a lossless 16-bit depth per RGB channel.

- **Real and diverse defocus blur**. Unlike other existing datasets, our dataset provides real defocus blur and in-focus pairs indicative of real camera optics.

- **Varying scene contents**. To provide a wide range of object categories, we collect 500 pairs of unique indoor/outdoor scenes with a large variety of scene contents. Our dataset is also free of faces to avoid privacy issues.

The $f/4$ (blurry) and $f/22$ (sharp) image pairs are carefully imaged static scenes with the camera fixed on a tripod. To further avoid camera shake, the camera was controlled remotely to allow hands-free operation. Figure 5.5 shows an example of an image pair from our dataset. The left and right DP views of $\mathrm{I_B}$ are provided by the camera and denoted as $\mathrm{I_L}$ and $\mathrm{I_R}$ respectively. The ISO setting is fixed for each image pair. Figure 5.5 shows the DP L/R views for only image $\mathrm{I_B}$, because DP L/R views of $\mathrm{I_S}$ are visually identical due to the fact $\mathrm{I_S}$ is our all-in-focus ground truth.

5.4 Dual-Pixel Defocus Deblurring DNN (DPDNet)

Using our captured dataset, we trained a symmetric encoder-decoder CNN architecture with skip connections between the corresponding feature maps [95, 96]. Skip connections are widely used in encoder-decoder CNNs to combine various levels of feature maps. These have been found useful for gradient propagation and convergence acceleration that allow training of deeper networks [97, 98].

We adapt a U-Net-like architecture [96]. U-Net is a well-known network architecture for image-to-image translation tasks (e.g., our task in this chapter end-to-end defocus deblurring). Other than that, U-Net has been demonstrated in [16] with thorough experimentation to be good for image deblurring tasks in general. The main reason is that U-Net can offer a large receptive field that can cover large blur PSFs. We modified the traditional U-Net as follows: an input layer to take a 6-channel input cube (two DP views; each is a 3-channel sRGB image) and an output layer to generate a 3-channel output sRGB image; skip connections of the convolutional feature maps are passed to their mirrored convolutional layers without cropping in order to pass on more feature map detail; and the loss function is changed to be a mean squared error (MSE) since the original U-Net was introduced for a classification task that adapted cross-entropy loss (i.e., the

cross-entropy loss is not appropriate for our regression task).

The overall DNN architecture of our proposed DP deblurring method is shown in Figure 5.6. Our method reads the two DP images, $\mathrm{I_L}$ and $\mathrm{I_R}$, as a 6-channel cube, and processes them through the encoder, bottleneck, and decoder stages to get the final sharp image $\mathrm{I_S^*}$. There are four blocks in the encoder stage (E-Block 1–4) and in each block, two 3×3 convolutional operations are performed, each followed by a ReLU activation. Then a 2×2 max pooling is performed for downsampling. Although max pooling operations reduce the size of feature maps between E-Blocks, this is required to extend the receptive field size in order to handle large defocus blur. To reduce the chances of overfitting, two dropout layers are added, one before the max pooling operation in the fourth E-Block, and one dropout layer at the end of the network bottleneck, as shown in Figure 5.6. In the decoder stage, we also have four blocks (D-Block 1–4). For each D-Block, a 2×2 upsampling of the input feature map followed by a 2×2 convolution ($Up - conv$) is carried out instead of direct deconvolution in order to avoid checkerboard artifacts [99]. The corresponding feature map from the encoder stage is concatenated. Next, two 3×3 convolutions are performed, each followed by a ReLU activation. Afterwards, a 1×1 convolution followed by sigmoid activation is applied to output the final sharp image $\mathrm{I_S^*}$. The number of output filters is shown under each convolution layer for each block in Figure 5.6. The stride for all operations is 1 except for the max pooling operation, which has a stride of 2. The final sharp image $\mathrm{I_S^*}$ is, thus, predicted as follows:

$$\mathrm{I_S^*} = \mathrm{DPDNet}(\mathrm{I_L}, \mathrm{I_R}; \theta_{\mathrm{DPDNet}}), \tag{5.1}$$

where DPDNet is our proposed architecture, and θ_{DPDNet} is the set of weights and parameters.

Training procedure The size of input and output layers is set to $512 \times 512 \times 6$ and $512 \times 512 \times 3$, respectively. This is because we train not on the full-size images but on the extracted image patches. We adopt the weight initialization strategy proposed by He [100] and use the Adam optimizer [90] to train the model. The initial learning rate is set to 2×10^{-5}, which is decreased by half every 60 epochs. We train our model with mini-batches of size 5 using MSE loss between the output and the ground truth as follows:

$$\mathcal{L} = \frac{1}{n} \sum_{n} (\mathrm{I_S} - \mathrm{I_S^*})^2, \tag{5.2}$$

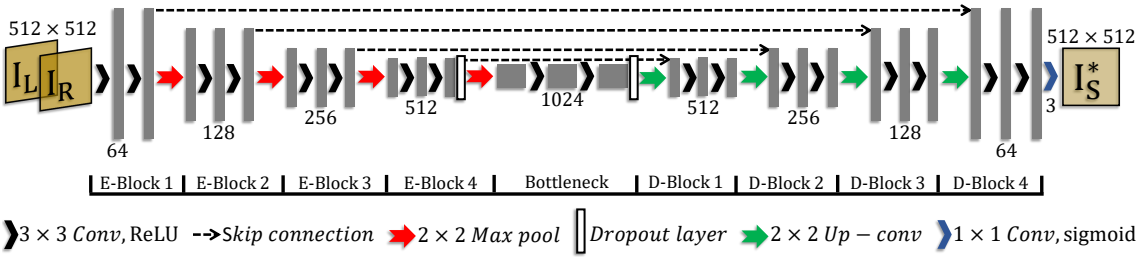


Figure 5.6: Our proposed DP deblurring architecture (DPDNet). Our method utilizes the DP images, I_L and I_R, for predicting the sharp image I_S^* through three stages: encoder (E-Blocks), bottleneck, and decoder (D-Blocks). The size of the input and output layers is shown above the images. The number of output filters is shown under the convolution operations for each block.

where n is the size of the image patch in pixels. During the training phase, we set the dropout rate to 0.4. All the models described in the subsequent sections are implemented using Python with the Keras framework [101] on top of TensorFlow [102] and trained with a NVIDIA TITAN X GPU. We set the maximum number of training epochs to 200.

5.5 Experimental Results

We first describe our data preparation procedure and then evaluation metrics used. This is followed by quantitative and qualitative results to evaluate our proposed method with existing deblurring methods. We also discuss the time analysis and test the robustness of our DP method against different aperture settings.

Data preparation Our dataset has an equal number of indoor and outdoor scenes. We divide the data into 70% training, 15% validation, and 15% testing sets. Each set has a balanced number of indoor/outdoor scenes. To prepare the data for training, we first downscale our images to be 1680×1120 in size. Next, image patches are extracted by sliding a window of size 512×512 with 60% overlap. We empirically found this image size and patch size to work well. An ablation study of different architecture settings is provided in Section 5.7. We compute the sharpness energy (i.e., by applying Sobel filter) of the in-focus image patches and sort them. We discard 30% of the patches that have the lowest sharpness energy. Such patches represent homogeneous regions, cause an ambiguity

Table 5.1: The quantitative results for different defocus deblurring methods. The testing on the dataset is divided into three scene categories: indoor, outdoor, and combined. The top result numbers are highlighted in green and the second top in blue. DPDNet-Single is our DPDNet variation that is trained with only a single blurred input. Our DPDNet that uses the two L/R DP views achieved the best results on all scene categories for all metrics. Note: the testing set consists of 37 indoor and 39 outdoor scenes.

Method	Indoor				Outdoor				Combined			
	PSNR ↑	SSIM ↑	MAE ↓	LPIPS ↓	PSNR ↑	SSIM ↑	MAE ↓	LPIPS ↓	PSNR ↑	SSIM ↑	MAE ↓	LPIPS ↓
EBDB [67]	25.77	0.772	0.040	0.297	21.25	0.599	0.058	0.373	23.45	0.683	0.049	0.336
DMENet [68]	25.50	0.788	0.038	0.298	21.43	0.644	0.063	0.397	23.41	0.714	0.051	0.349
JNB [70]	26.73	0.828	0.031	0.273	21.10	0.608	0.064	0.355	23.84	0.715	0.048	0.315
Our DPDNet-Single	26.54	0.816	0.031	0.239	22.25	0.682	0.056	0.313	24.34	0.747	0.044	0.277
Our DPDNet	**27.48**	**0.849**	**0.029**	**0.189**	**22.90**	**0.726**	**0.052**	**0.255**	**25.13**	**0.786**	**0.041**	**0.223**

associated to the amount of blur, and adversely affect the DNNs training, as found in [69].

Evaluation metrics Results are reported on traditional signal processing metrics – namely, peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM) [103], and mean absolute error (MAE). We also incorporate the recent learned perceptual image patch similarity (LPIPS) proposed by [104]. The LPIPS metric is correlated with human perceptual similarity judgments as a perceptual metric for low-level vision tasks, such as enhancement and image deblurring.

Quantitative results We compare our DPDNet with the following three methods: the edge-based defocus blur (EBDB) [67], the defocus map estimation network (DMENet) [68], and the just noticeable blur (JNB) [70] estimation. These methods accept only a single image as input – namely, I_B – and estimate the defocus map in order to use it to guide the deblurring process. The EBDB [67] and JNB [70] are not learning-based methods. We test them directly on our dataset using I_B as input. The EBDB uses a combination of non-blind deblurring methods proposed in [74, 105], and for a fair comparison, we contacted the authors for their deblurring settings and implementation. The JNB method uses the non-blind defocus deblurring method from [73].

For the deep-learning-based method (i.e., DMENet [68]), the method requires the ground truth defocus map for training. In our dataset, we do not have this ground truth

Figure 5.7: Qualitative comparisons of different deblurring methods. The first row is the input image that has a spatially varying blur, and the last row is the corresponding ground truth sharp image. The rows in between are the results of different methods. We also present zoomed-in cropped patches in green and red boxes. Our DPDNet method notably outperforms other methods in terms of deblurring quality.

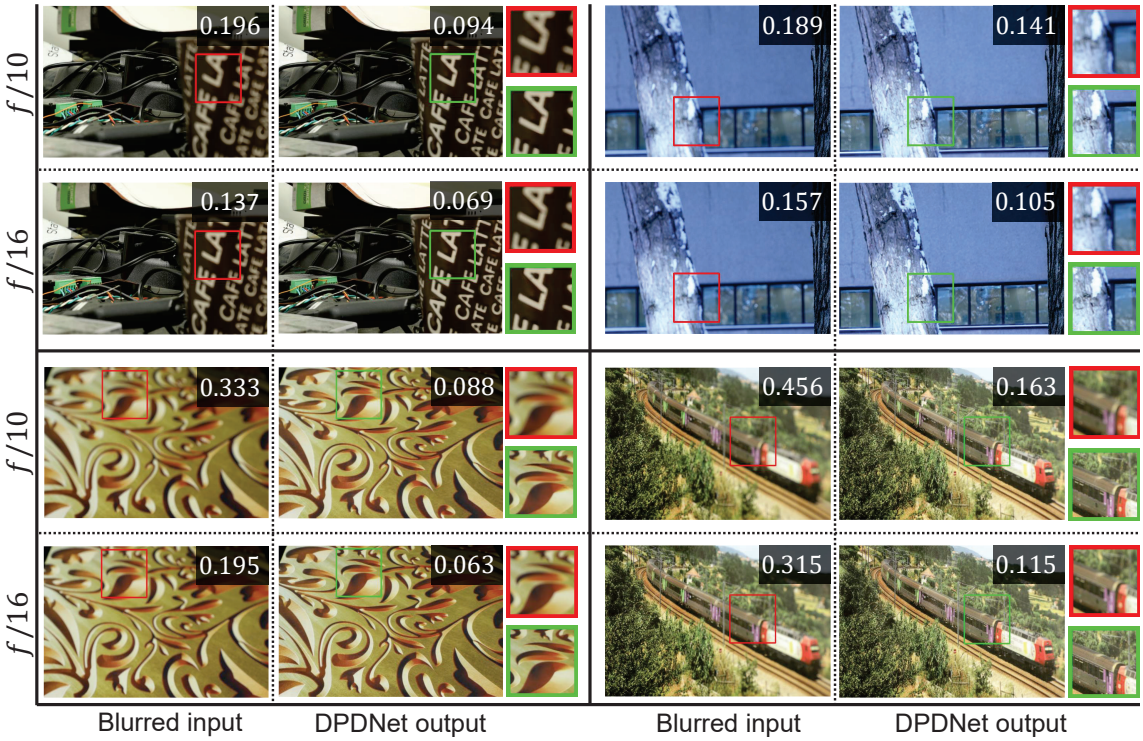


Figure 5.8: Examining DPDNet's robustness to different aperture settings. Four scenes are presented; each has two different apertures. In each scene, the left-hand image is the blurred one, I_B, and the right-hand image is the deblurred one, I_S^*, computed by our DPDNet. The number shown on each image is the LPIPS measure compared with the ground truth I_S. Zoomed-in cropped patches are also provided. Even though our training data was on blurry examples with an $f/4$ aperture, our DPDNet method is able to generalize well to different aperture settings.

defocus map and provide only the sharp image, since our approach in this work is to solve directly for defocus deblurring. Therefore, we tested the DMENet on our dataset using I_B as input without retraining. For deblurring, DMENet adopts a non-blind deconvolution algorithm proposed by [74]. Our results are compared against code provided by the authors. Unfortunately, the methods in [66, 69] do not have the deblurring code available for comparison.

To show the advantage of utilizing DP data for defocus deblurring, we introduce a variation of our DPDNet that accepts only a single input (i.e., I_B) and uses exactly the same architecture settings along with the same training procedure as shown in Fig 5.6. We

refer to this variation as DPDNet-Single in Table 5.1. Our proposed architecture is fully convolutional, which enables testing any image size during the testing phase. Therefore, all the subsequent results are reported on the testing set using the full image for all methods. Table 5.1 reports our findings by testing on three scene categories: indoor, outdoor, and combined. Top result numbers are highlighted in green and the second top ones in blue. Our DPDNet method has a notably better deblurring ability based on all metrics for all testing categories. Furthermore, DP data is the key that made our DPDNet method outperforms others, especially the single image input one (i.e., DPDNet-Single), in which it has exactly the same architecture but does not utilize DP views. Interestingly, all methods have better deblurring results for indoor scenes, due to the fact that outdoor scenes tend to have larger depth variations, and thereby more defocus blur.

Qualitative results In Figure 5.7, we present the qualitative results of different defocus deblurring methods. The first row shows the input image with a spatially varying defocus blur; the last row shows the corresponding ground truth sharp image. The rows in between present different methods, including ours. This figure also shows two zoomed-in cropped patches in green and red to further illustrate the difference visually. From the visual comparison with other methods, our DPDNet has the best deblurring ability and is quite similar to the ground truth. EBDB [67], DMENet [68], and JNB [70] are not able to handle spatially varying blur with almost unnoticeable difference with the input image. EBDB [67] tends to introduce some artifacts in some cases. Our single image method (i.e., DPDNet-Single) has better deblurring ability compared to other traditional deblurring methods, but it is not at the level of our method that utilizes DP views for deblurring. Our DPDNet, as shown visually, is effective in handling spatially varying blur. For example, in the second row, the image has a part that is in focus and another is not; our DPDNet method is able to determine the deblurring amount required for each pixel, in which the in-focus part is left untouched.

Time analysis We examine evaluating different defocus deblurring methods based on the time required to process a testing image of size 1680×1120 pixels. Our DPDNet directly computes the sharp image in a single pass, whereas other methods [67, 68, 70] use two passes: (1) defocus map estimation and (2) non-blind deblurring based on the estimated defocus map.

Table 5.2: Time analysis of different defocus deblurring methods using CPU and GPU when applicable. The last column is the total time required to process a testing image of size 1680×1120 pixels. Our DPDNet is about 75 times faster compared to the second-best method (i.e., DMENet) when tested using CPU.

Method	Time-CPU (Sec) ↓			Time-GPU (Sec) ↓		
	Defocus map estimation	Defocus deblurring	Total	Defocus map estimation	Defocus deblurring	Total
EBDB [67]	57.2	872.5	929.7	N/A	N/A	N/A
DMENet [68]	9.3	612.4	613.7	1.3	N/A	N/A
JNB [70]	605.4	237.7	843.1	N/A	N/A	N/A
Our DPDNet	**0**	**8.2**	**8.2**	**0**	**0.5**	**0.5**

Non-learning-based methods (i.e., EBDB [67] and JNB [70]) do not utilize the GPU and use only the CPU (i.e., running them on GPUs will not speed up the process). For the deep-learning method (i.e., DMENet [68]), it utilizes the GPU for the first pass; however, the deblurring routine is applied on a CPU. This time evaluation is performed using Intel Core i7-6700 CPU and NVIDIA TITAN X GPU. Our DPDNet operates in a single pass and can process the testing image of size 1680×1120 pixels about 75 times faster compared to the second-best method (i.e., DMENet) when running on CPU as shown in Table 5.2.

Robustness to different aperture settings In our dataset, the image pairs are captured using aperture settings corresponding to f-stops $f/22$ and $f/4$. Recall that $f/4$ results in the greatest DoF and thus most defocus blur. Our DPDNet is trained on diverse images with many different depth values; thus, our training data spans the worst-case blur that would be observed with any aperture settings. To test the ability of our DPDNet in generalizing for scenes with different aperture settings, we capture image pairs with aperture settings $f/10$ and $f/16$ for the blurred image and again $f/22$ for the corresponding ground truth image. Our DPDNet is applied to these less blurred images. Figure 5.8 shows the results for four scenes, where each scene's image has its LPIPS measure compared with the ground truth. For better visual comparison, Figure 5.8 provides zoomed-in patches that are cropped from the blurred input (red box) and the deblurred one (green box). These results show that our DPDNet is able to deblur scenes with different aperture settings that have not been used during training.

5.6 DPNet Performance for a Smartphone DP Sensor

In this section, we test our DPNet on images captured with a smartphone. As we mentioned in Section5.3, there are two camera manufacturers that provide DP data, namely, Google Pixel 3 and 4 smartphones and Canon EOS 5D Mark IV DSLR. The smartphone camera currently has limitations that make it challenging to train the DPNet with. First, the Google Pixel smartphone cameras do not have adjustable apertures, so we are unable to capture corresponding "sharp" images using a small aperture as we did with the Canon camera. Second, the data currently available from the Pixel smartphones are not full-frame, but are limited to only one of the Green channels in the raw-Bayer frame. Finally, the smartphone has a very small aperture so most images do not exhibit defocus blur. In fact, many smartphone cameras synthetically apply defocus blur to produce the shallow DoF effect.

As a result, the experiments here are provided to serve as a proof of concept that our method should generalize to other DP sensors. To this end, we examined DP images available in the dataset from [25] to find images exhibiting defocus blur. The L/R views of these images are available in the "animated_DP_examples" directory—located at the same directory as this pdf file.

To use our DPNet, we replicate the single green channel to be 3-channel image to match our DPNet input. Figure 5.9 shows the deblurring results on images captured by Pixel camera. The image on the left is the input combined image and the image on the right is the deblurred one using our DPNet. Note that the Pixel android application, used to extract DP data, does not provide the combined image [93]. To obtain it, we average the two views. Figure 5.9 visually demonstrates that our DPNet is able to generalize and deblur for images that are captured by the smartphone camera. Because it is not possible to adjust aperture on the smartphone camera to capture a ground truth image, we cannot report quantitative numbers. The results of two full images are shown in Figure 5.10.

5.7 Ablation Study

In this section, we provide an ablation study of different variations in training our DPNet with: (1) an extra input image (Section 5.7.1), (2) less E-Blocks and D-Blocks (Section 5.7.2), (3) different input sizes (Section 5.7.3), (4) different ratios of homogeneous

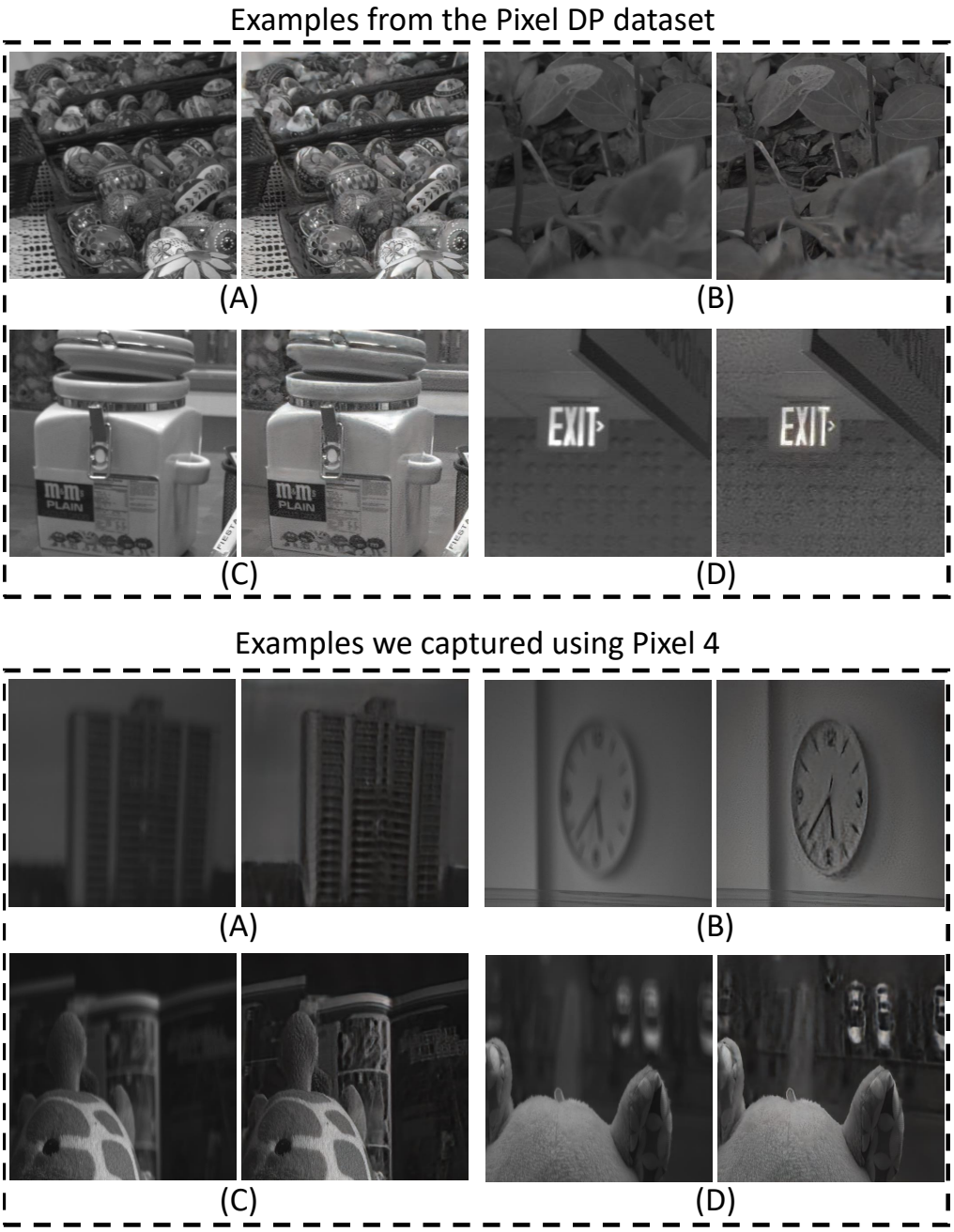
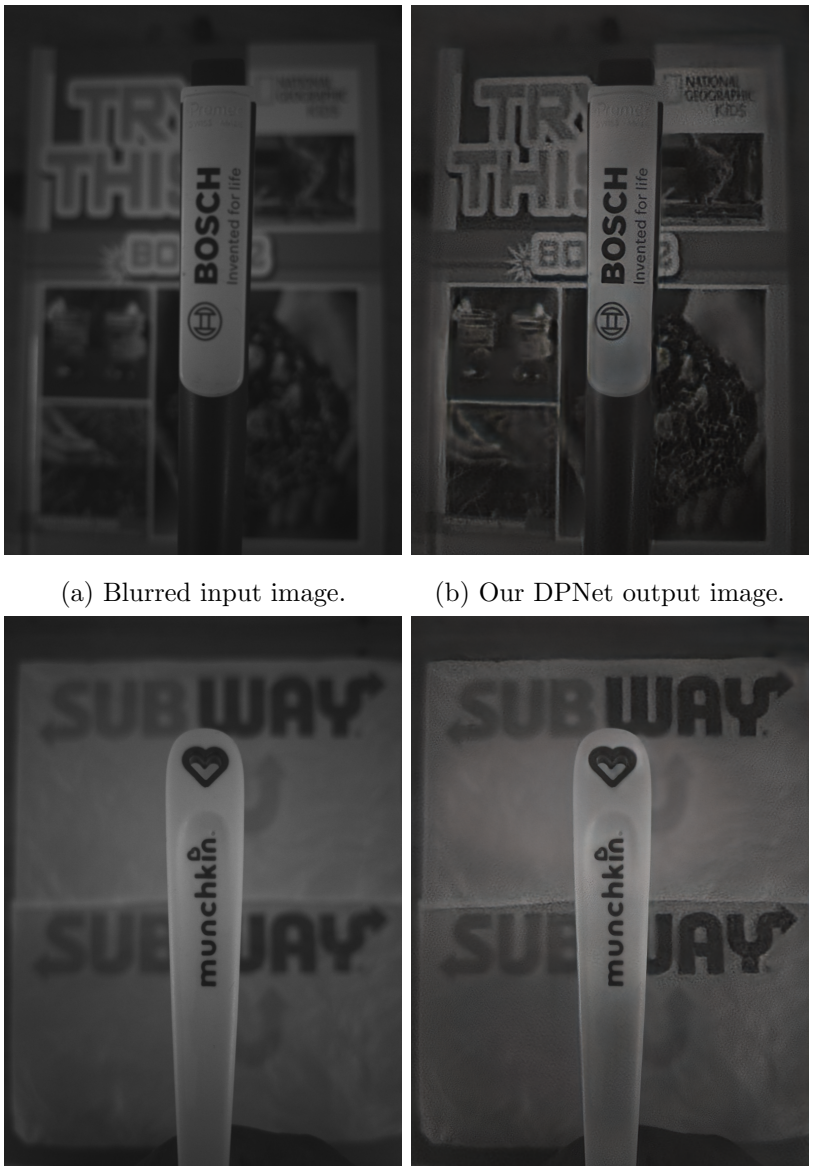


Figure 5.9: The results of using our DPNet to deblur images captured by Pixel smartphone camera. The image on the left is the combined input image with defocus blur and the one on the right is the deblurred one. Our DPNet is able to generalize well for images captured by a smartphone camera.



(a) Blurred input image. (b) Our DPNet output image.

(c) Blurred input image. (d) Our DPNet output image.

Figure 5.10: Qualitative deblurring results using our DPNet for images captured by a smartphone camera.

region filtering (Section 5.7.4), and (5) different data types (Section 5.7.5).

5.7.1 DPNet With Extra Input Image

As described in Section 5.4, our DPNet takes the two dual-pixel L/R views, I_L and I_R, as inputs to estimate the sharp image I_S^*. In our dataset, in addition to the L/R views, we also provide the corresponding combined image I_B that would be outputted by the camera. In this section, we examine training our DPNet with all three images, namely I_L, I_R, and I_B. We refer to this variation as DPNet(I_L, I_R, I_B).

Table 5.3 shows the results of the three-input DPNet, DPNet(I_L, I_R, I_B), vs. the two-input one, DPNet(I_L, I_R). The results of all metrics are quite similar with a slight difference. Our conclusion is that training and testing the DPNet with the extra input I_B provides no noticeable improvement. Such results are expected, since I_B is a combination of I_L and I_R.

Table 5.3: DPNet with extra input image. The quantitative results of DPNet(I_L, I_R, I_B) vs. DPNet($I_L, I_R, $) using four metrics. The testing on the dataset is divided into three scene categories: indoor, outdoor, and combined. The best results are in bold numbers. The results of DPNet(I_L, I_R, I_B) and DPNet($I_L, I_R, $) are quite similar with a slight difference. Note: the testing set consists of 37 indoor and 39 outdoor scenes.

Method	**Indoor**				**Outdoor**				**Combined**			
	PSNR ↑	SSIM ↑	MAE ↓	LPIPS ↓	PSNR ↑	SSIM ↑	MAE ↓	LPIPS ↓	PSNR ↑	SSIM ↑	MAE ↓	LPIPS ↓
DPDNet(I_L, I_R, I_B)	27.32	0.842	**0.029**	0.191	**22.94**	0.723	**0.052**	0.257	25.07	0.781	**0.041**	0.225
DPDNet(I_L, I_R)	**27.48**	**0.849**	**0.029**	**0.189**	22.90	**0.726**	**0.052**	**0.255**	**25.13**	**0.786**	**0.041**	**0.223**

5.7.2 DPNet With Less Blocks

In this section, we train a "lighter" version of our DPNet with less E-Blocks and D-Blocks. This is done by reducing E-Block 1 and D-Block 4. We refer to this light version as DPNet-Light. In Table 5.4, we provide a comparison of DPNet-Light and our full DPNet.

Table 5.4 shows that our full DPNet has a better performance compared to the lighter one. Nevertheless, the sacrifice in performance is not notable, which implies that the DPNet-Light could be an option for environments with limited computational resources.

Table 5.4: DPNet with less blocks. The quantitative results of DPNet-Light vs. our full DPNet using four metrics. The testing on the dataset is divided into three scene categories: indoor, outdoor, and combined. The best results are in bold numbers. Our full DPNet has the best results on all metrics for different categories. Nevertheless, DPNet-Light can operate with less computational power and produce acceptable deblurring results. Note: the testing set consists of 37 indoor and 39 outdoor scenes.

Method	Indoor				Outdoor				Combined			
	PSNR ↑	SSIM ↑	MAE ↓	LPIPS ↓	PSNR ↑	SSIM ↑	MAE ↓	LPIPS ↓	PSNR ↑	SSIM ↑	MAE ↓	LPIPS ↓
DPDNet-Light	27.08	0.824	0.030	0.225	22.81	0.701	0.053	0.309	24.89	0.761	0.042	0.268
DPDNet	**27.48**	**0.849**	**0.029**	**0.189**	**22.90**	**0.726**	**0.052**	**0.255**	**25.13**	**0.786**	**0.041**	**0.223**

5.7.3 DPNet With Different Input Sizes

Our DPNet is a fully convolutional network. This facilitates training with different input patch sizes with no change required in the network architecture. As such, we consider training with two different patch sizes, namely 256×256 pixels and 512×512 pixels referred to as DPNet_{256} and DPNet_{512}, respectively.

Table 5.5 shows that the two different input sizes perform similarly. Particularly, input patch size does not change the performance drastically as long as it is larger than the blur size.

Table 5.5: DPNet with different input sizes. The quantitative results of DPNet_{256} vs. DPNet_{512} using four metrics. The testing on the dataset is divided into three scene categories: indoor, outdoor, and combined. The best results are in bold numbers. Both input sizes perform on par, in which the patch size does not change the performance drastically as long as it is larger than the blur size. Note: the testing set consists of 37 indoor and 39 outdoor scenes.

Method	Indoor				Outdoor				Combined			
	PSNR ↑	SSIM ↑	MAE ↓	LPIPS ↓	PSNR ↑	SSIM ↑	MAE ↓	LPIPS ↓	PSNR ↑	SSIM ↑	MAE ↓	LPIPS ↓
DPDNet_{256}	27.28	0.847	**0.029**	0.195	22.86	**0.734**	**0.050**	0.257	25.01	**0.789**	**0.040**	0.227
DPDNet_{512}	**27.48**	**0.849**	**0.029**	**0.189**	**22.90**	0.726	0.052	**0.255**	**25.13**	0.786	0.041	**0.223**

5.7.4 DPNet With Different Filtering Ratios

Homogeneous patches are inherently ambiguous in terms of incurred blur size, and do not provide useful information for network training [69]. As a result, filtering homogeneous patches can be beneficial to the trained network. In this section, different filtering ratios are examined including: 0%, 15%, 30%, and 45%; we refer to them as $\text{DPNet}_{0\%}$, $\text{DPNet}_{15\%}$, $\text{DPNet}_{30\%}$, $\text{DPNet}_{45\%}$, respectively.

In Table 5.6, we present the results of different filtering ratios. The 30% filtering is a reasonable ratio that has the best quantitative results. Therefore, we filter 30% of the extracted image patches based on the sharpness energy to train our proposed DPNet as described in Section 5.5.

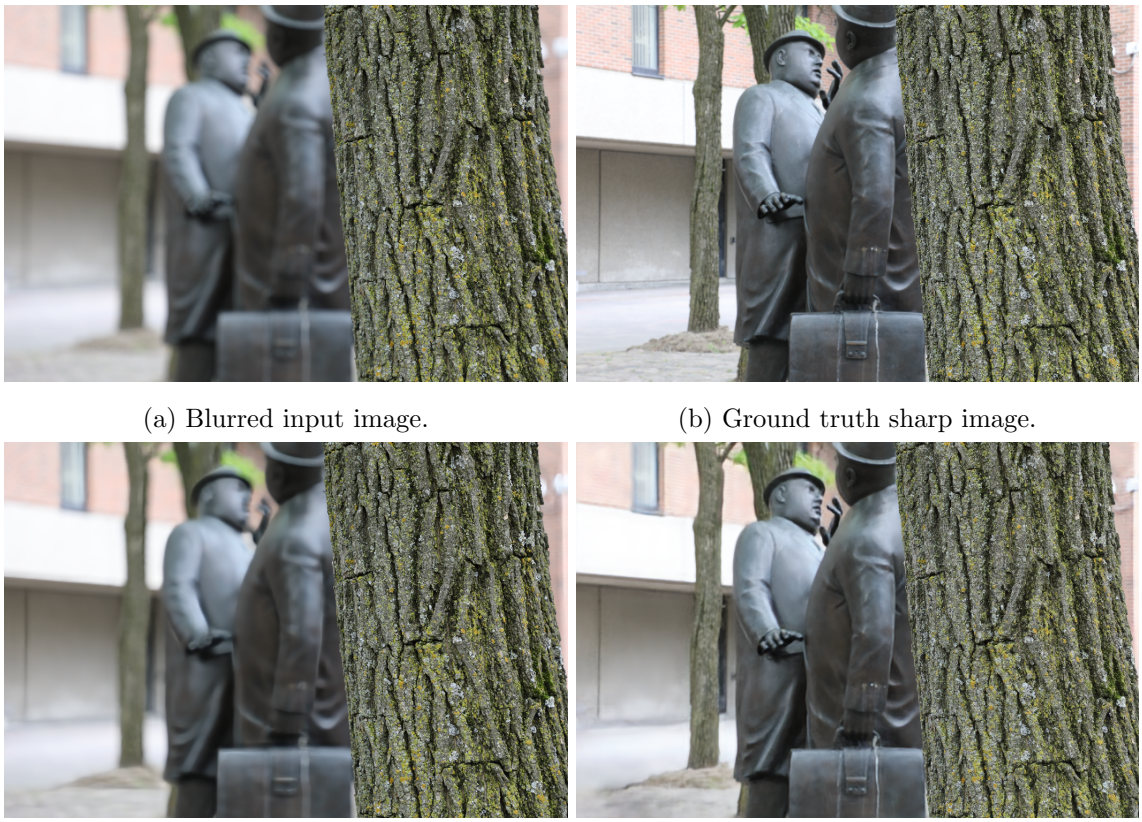
Table 5.6: DPNet with different filtering ratios. The quantitative results of $\text{DPNet}_{0\%}$ vs. $\text{DPNet}_{15\%}$ vs. $\text{DPNet}_{30\%}$ vs. $\text{DPNet}_{45\%}$ using four metrics. The testing on the dataset is divided into three scene categories: indoor, outdoor, and combined. The best results are in bold numbers. The 30% filtering is a reasonable ratio that has the best quantitative results and , thus, we pick it as a filtering ratio for our proposed framework. Note: the testing set consists of 37 indoor and 39 outdoor scenes.

Method	**Indoor**				**Outdoor**				**Combined**			
	PSNR ↑	SSIM ↑	MAE ↓	LPIPS ↓	PSNR ↑	SSIM ↑	MAE ↓	LPIPS ↓	PSNR ↑	SSIM ↑	MAE ↓	LPIPS ↓
$\text{DPDNet}_{0\%}$	27.21	0.838	0.030	0.205	22.86	0.721	**0.051**	0.275	24.98	0.778	**0.041**	0.241
$\text{DPDNet}_{15\%}$	27.19	0.840	**0.029**	0.194	**22.94**	0.721	0.052	**0.254**	25.01	0.779	**0.041**	0.225
$\text{DPDNet}_{30\%}$	**27.48**	**0.849**	**0.029**	**0.189**	22.90	**0.726**	0.052	0.255	**25.13**	**0.786**	**0.041**	**0.223**
$\text{DPDNet}_{45\%}$	27.21	0.839	0.030	0.194	22.90	0.724	**0.051**	0.258	25.00	0.780	**0.041**	0.227

5.7.5 DPNet With Different Data Types

Our dataset provides high-quality images that are processed to an sRGB encoding with a lossless 16-bit depth per RGB channel. Since we are targeting dual-pixel information which would be obtained directly in the camera's hardware, in a real hardware implementation we would expect to have such high bit-depth images. However, since most standard encodings still rely on 8-bit image, we provide a comparison of training our DPNet with 8-bit (DPNet_{8-bit}) and 16-bit (DPNet_{16-bit}) input data type.

Based on the numbers in Table 5.7, DPNet_{16-bit} has a slightly better performance. In



(a) Blurred input image.

(b) Ground truth sharp image.

(c) SRNet [2] output image.

(d) Our DPNet output image.

Figure 5.11: Qualitative deblurring results using SRNet [2] and our DPNet.

particular, it has a lower LPIPS distance for all categories. As a result, training with 16-bit images is helpful due to the extra information embedded in, and is more representative of the hardware's data. One more note, the PSNR of DPNet$_{8-bit}$ is slightly higher and it could be that the 16-bit images with more precision are sensitive to MSE measure and, thus, affecting the resultant PSNR.

5.8 Use Cases

As discussed in Section 5.1, we described how defocus blur is related to the size of the aperture used at capture time. The size of the aperture is often dictated by the desired exposure which is a factor of aperture, shutter speed, and ISO setting. As a result, there is a trade-off between image noise (from ISO gain), motion blur (shutter speed), and defocus

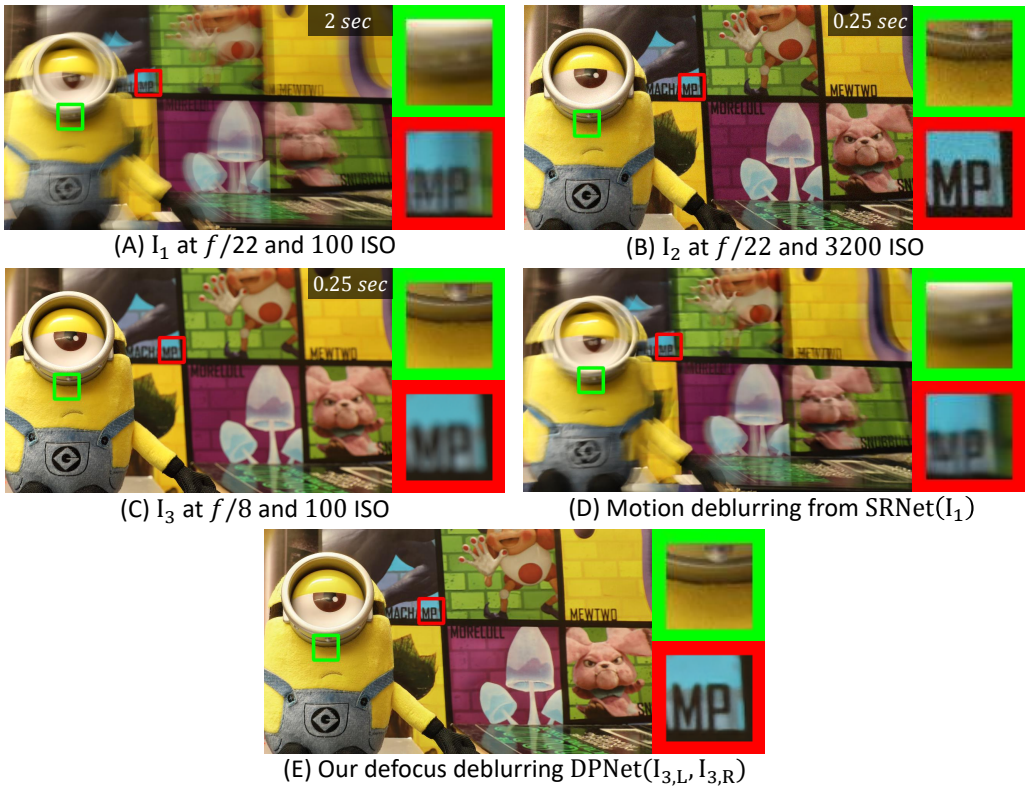
Table 5.7: DPNet with different data types. The quantitative results of DPNet_{8-bit} vs. DPNet_{16-bit} using four metrics. The testing on the dataset is divided into three scene categories: indoor, outdoor, and combined. The best results are in bold numbers. DPNet_{16-bit} has a slightly better performance, in which it has a lower LPIPS distance for all categories. Note: the testing set consists of 37 indoor and 39 outdoor scenes.

Method	Indoor				Outdoor				Combined			
	PSNR ↑	SSIM ↑	MAE ↓	LPIPS ↓	PSNR ↑	SSIM ↑	MAE ↓	LPIPS ↓	PSNR ↑	SSIM ↑	MAE ↓	LPIPS ↓
DPDNet$_{8-bit}$	27.37	0.834	**0.029**	0.196	**23.10**	0.723	**0.052**	0.258	**25.18**	0.777	**0.041**	0.228
DPDNet$_{16-bit}$	**27.48**	**0.849**	**0.029**	**0.189**	22.90	**0.726**	**0.052**	**0.255**	25.13	**0.786**	**0.041**	**0.223**

blur (aperture). This trade off is referred to as the exposure triangle. In this section, we show some common cases, where defocus deblurring is required.

Moving camera Global motion blur is more likely to occur with the moving cameras like hand-held cameras (I_1 in Figure 5.12-A). One way to handle motion blur is to set a fast shutter speed and this can be done by either increasing the image gain (i.e., ISO) or the aperture size. However, higher ISO can introduce noise as stated in [94] (Figure 5.12-B), and wider aperture can introduce undesired defocus blur as shown in I_3 (Figure 5.12-C). For such case, we offer two solutions: apply motion deblurring method SRNet [2] on I_1 (result shown in Figure 5.12-D) or apply our defocus deblurring method on I_3 (result shown in Figure 5.12-E). Our defocus deblurring method is able to obtain sharper and cleaner image as demonstrated in Figure 5.12-E.

Moving object In this scenario, we have a stationary camera, with a scene object that is moving (i.e., Newton's cradle in Figure 5.13). Figure 5.13-A shows an image with motion blur, in which the object speed is higher than the shutter speed. In Figure 5.13-B, the ISO is increased in order to make the shutter speed faster, nevertheless, the pendulum speed remains the fastest and the motion blur is pronounced. Another way to increase the shutter speed is to open the aperture wider as shown in Figure 5.13-C and this setting handles the motion blur. However, capturing at wider aperture introduces the undesired defocus blur. To get a sharper image, we can use the motion deblurring method SRNet [2] to deblur I_1 (result shown in Figure 5.13-D) and I_2 (result shown in Figure 5.13-E), or apply our defocus deblurring method on I_3 (result shown in Figure 5.13-F). Our defocus



(A) I_1 at $f/22$ and 100 ISO

(B) I_2 at $f/22$ and 3200 ISO

(C) I_3 at $f/8$ and 100 ISO

(D) Motion deblurring from SRNet(I_1)

(E) Our defocus deblurring DPNet($I_{3,L}$, $I_{3,R}$)

Figure 5.12: Image noise, motion and defocus blur relation with a moving camera. The number shown on each image is the shutter speed. Zoomed-in cropped patches are also provided. (A) shows an image I_1 suffers from motion blur. (B) shows an image I_2 fixes the motion blur by increasing the ISO, however, I_2 has more noise. (C) shows another image I_3 handles the motion blur by increasing the aperture size, nevertheless, I_3 suffers from defocus blur. (D) shows the results of deblurring I_1 using the motion deblurring method SRNet [2]. The image in (E) is the sharp and clean image obtained using our DPNet to deblur I_3.

deblurring method is able to obtain sharper image compared to motion deblurring method as demonstrated in Figure 5.13-F.

5.9 Applications

Image blur can have a negative impact on some computer vision tasks, as found in [106]. Here we investigate defocus blur effect on two common computer vision tasks – namely,

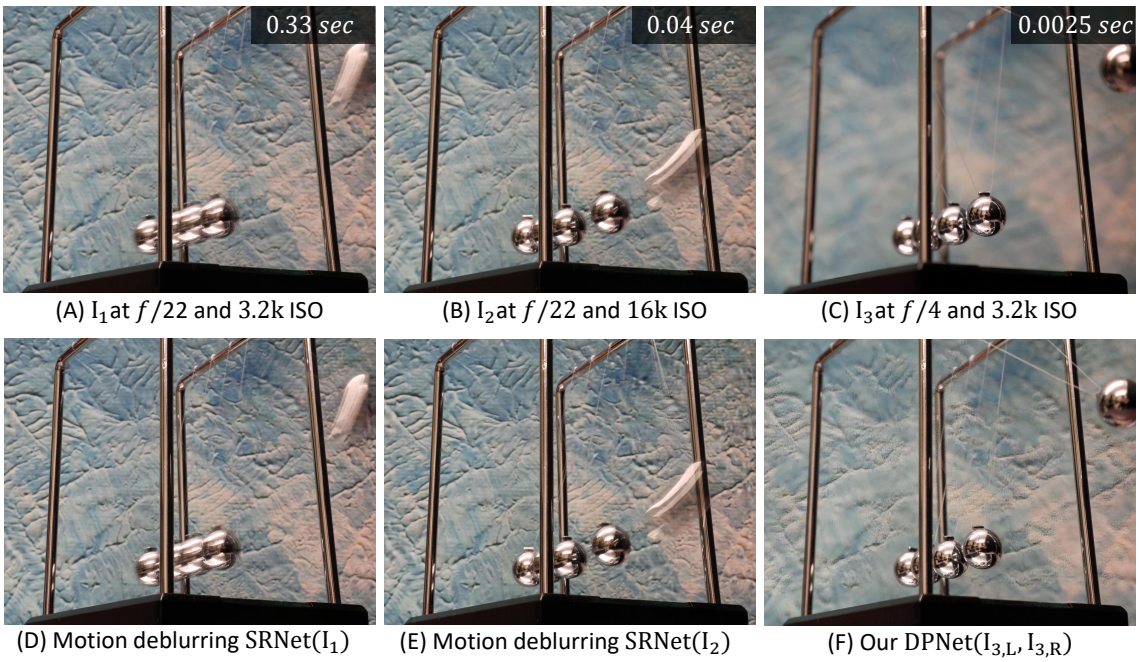


Figure 5.13: Motion and defocus blur relation with a moving object. The number shown on each image is the shutter speed. (A) shows an image I_1 has a moving object that suffers from motion blur. Image I_2 in (B) tries to fix the motion blur by increasing the ISO, but the motion blur is still pronounced. I_3 in (C) handles the motion blur by setting the aperture wide, nevertheless, it introduces defocus blur. (D) and (E) show the results of deblurring I_1 and I_2, respectively, using the motion deblurring method SRNet [2]. The image in (F) is sharp and obtained by drblurring I_3 using our DPNet.

image segmentation and monocular depth estimation.

Image segmentation The first two columns in Figure 5.14 demonstrate the negative effect of defocus blur on the task of image segmentation. We use the PSPNet segementation model from [107], and test two images: one is the blurred input image I_B and another is the deblurred one I_S^* using our DPDNet deblurring model. The segmentation results are affected by I_B – only the foreground tree was correctly segmented. PSPNet assigns cyan color to unknown categories, where a large portion of I_B is segmented as unknown. On the other hand, the segmentation results of I_S^* are much better, in which more categories are segmented correctly. With that said, image DoF deblurring using our DP method can be beneficial for the task of image segmentation.

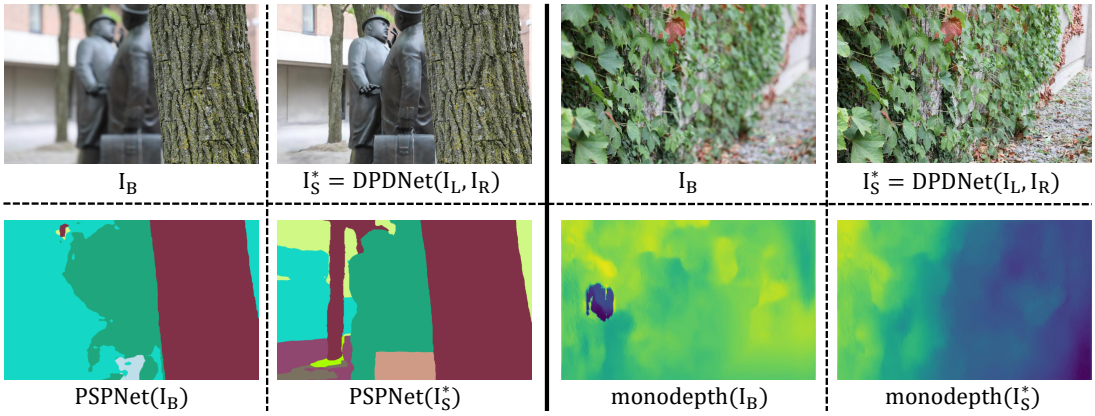


Figure 5.14: The effect of defocus blur on some computer vision tasks. The first two columns show the image segmentation results using the PSPNet [107] segmentation model. The segmentation results are affected by the blurred image I_B, where a large portion is segmented as unknown in cyan. The last two columns show the results of the monocular depth estimation using the monodepth model from [108]. The depth estimation is highly affected by the defocus blur and produced wrong results. Deblurring I_B using our DP deblurring method has notably improved the results for both tasks.

Monocular depth estimation The monocular depth estimation is the task of estimating scene depth using a single image. In the last two columns of Figure 5.14, we show the direct effect of defocus blur on this task. We use the monodepth model from [108] to test the two images I_B and I_S^* in order to examine the change in performance. The result of monodepth is affected by the defocus blur, in which the depth map estimated is completely wrong. In contrast, the result of monodepth has been notably improved after testing with the deblurred input image using our DPDNet deblurring model. Therefore, deblurring images using our DPDNet can be useful for the task of monocular depth map estimation.

5.10 Defocus and Motion Blur Discussion

One may be curious if motion blur methods can be used to address the defocus blur problem. While defocus and motion blur both produce a blurring of the underlying latent image, the physical image formation process of these two types of blur are different. Therefore, comparing with methods that solve for motion blur is not expected to give good

results. However, for a validity check, we tested the scale recurrent network (SRNet) for motion deblurring in [2] using our testing set. This method achieved an average LPIPS of 0.452 and PSNR of 20.12, which is lower than all other existing methods that solve for defocus deblurring. Figure 5.11 shows results of applying motion deblurring network SRNet [2] to input image from our dataset.

5.11 Summary

We have presented a novel approach to reduce the effect of defocus blur present in images captured with a shallow DoF. Our approach leverages the DP data that is available in most modern camera sensors but currently being ignored for other uses. We show that the DP images are highly effective in reducing DoF blur when used in a DNN framework. As part of this effort, we have captured a new image dataset consisting of blurred and sharp image pairs along with their DP images. Experimental results show that leveraging the DP data provides state-of-the-art quantitative results on both signal processing and perceptual metrics. We also demonstrate that our deblurring method can be beneficial for other computer vision tasks.

We demonstrated the utility of reducing defocus blur through the use cases and applications presented in this chapter. While defocus deblurring is important, it receives much lower attention from the industry and research community compared to motion deblurring (see Figure 1.3). To further encourage the research related to defocus deblurring, we organized a challenge in conjunction with CVPR 2021 [109] (More details about this challenge are provided in Appendix A). To conclude, we believe our captured dataset and DP-based method are useful for the research community and will help spur additional ideas about both defocus deblurring and applications that can leverage data from DP sensors. The dataset, code, and trained models are available at `https://github.com/Abdullah-Abuolaim/defocus-deblurring-dual-pixel`.

Chapter 6

Extending Camera DoF for Video Data

Our work in Chapter 5 has shown promising results on data-driven defocus deblurring using the two-image views available on modern DP sensors. One significant challenge in this line of research is access to DP data. Despite many cameras having DP sensors, only a limited number provide access to the low-level DP sensor images. In addition, capturing training data for defocus deblurring involves a time-consuming and tedious setup requiring the camera's aperture to be adjusted (Section 5.3). Some cameras with DP sensors (e.g., smartphones) do not have adjustable apertures, further limiting the ability to produce the necessary training data. We address the data capture bottleneck by proposing a procedure to generate realistic DP data synthetically. Our synthesis approach mimics the optical image formation found on DP sensors and can be applied to virtual scenes rendered with standard computer software. Leveraging these realistic synthetic DP images, we introduce a recurrent convolutional network (RCN) architecture that improves deblurring results and is suitable for use with single-frame and multi-frame data (e.g., video) captured by DP sensors. Finally, we show that our synthetic DP data is useful for training DNN models targeting video deblurring applications where access to DP data remains challenging.

6.1 Introduction

Defocus blur occurs in scene regions captured outside the camera's depth of field (DoF). Although the effect can be intentional (e.g., the bokeh effect in portrait photos), in many

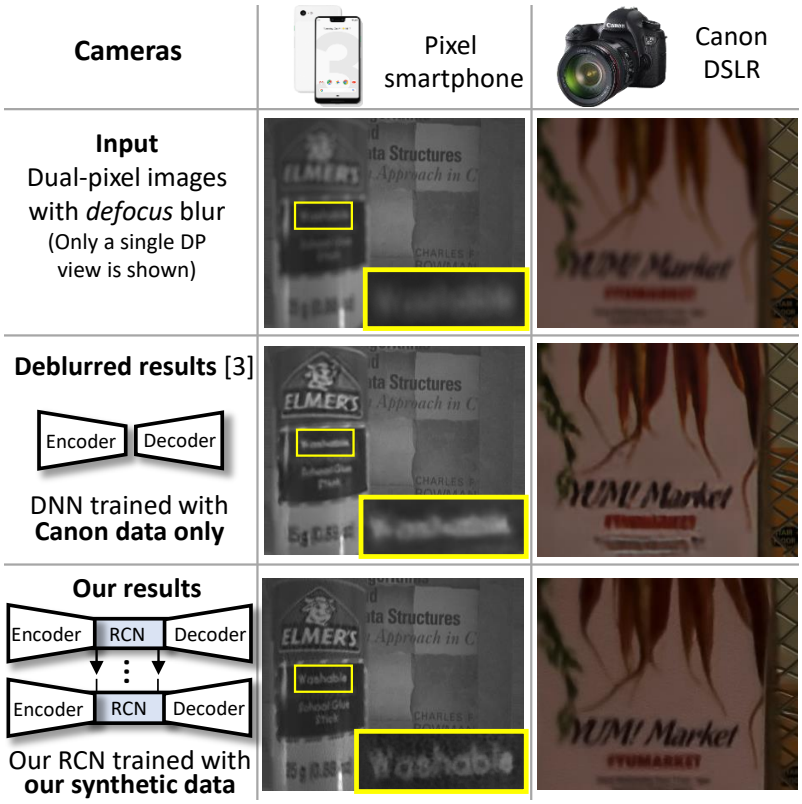


Figure 6.1: Deblurring results on images from a Pixel 4 smartphone and a Canon 5D Mark IV. **Third row:** results of the DNN proposed in [3] trained with DP data from the Canon camera. **Last row:** results from our proposed network trained on *synthetically generated data only*.

cases defocus blur is undesired as it impacts image quality due to the loss of sharpness of image detail (e.g., Figure 6.1, second row). Recovering defocused image details is challenging due to the spatially varying nature of the defocus point spread function (PSF) [71,72], which not only is scene depth dependent, but also varies based on the camera aperture, focal length, focus distance, radial distortion, and optical aberrations. Most existing deblurring methods [66–70] approach the defocus deblurring problem by first estimating a defocus image map. The defocus map is then used with an off-the-shelf non-blind deconvolution method (e.g., [73, 74]). This strategy to defocus deblurring is greatly limited by the accuracy of the estimated defocus map and results in a very slow methods due to the two-stage approach.

Recently, our work in [3] (Chapter 5) proposed an interesting approach to the defocus

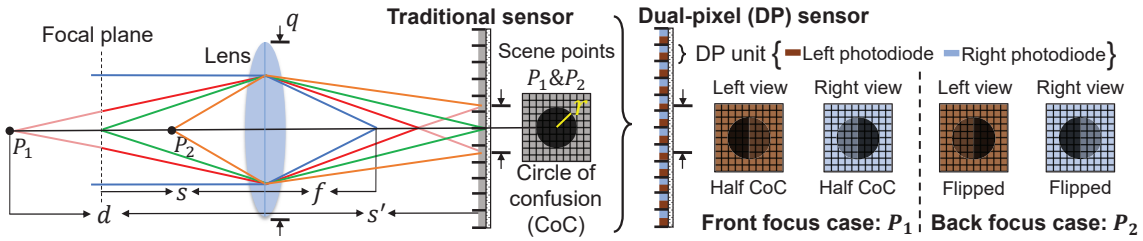


Figure 6.2: Thin lens model illustration and dual-pixel image formation. The circle of confusion (CoC) size is calculated for a given scene point using its distance from the lens, camera focal length, and aperture size. On the two dual-pixel views, the half-CoC PSF flips if the scene point is in front or back of the focal plane.

deblurring problem by leveraging information available on DP sensors found on most modern cameras. The DP sensor was originally designed to facilitate auto-focusing [11,14]; however, researchers have found DP sensors to be useful in broader applications, including depth map estimation [4,25,26], reflection removal [5], and synthetic DoF [27]. DP sensors consist of two photodiodes at each pixel location effectively providing the functionality of a simple two-sample light-field camera (Figure 6.2, DP sensor). Light rays coming from scene points within the camera's DoF will have no difference in phase, whereas light rays from scene points outside the camera's DoF will have a relative shift that is directly correlated to the amount of defocus blur. Recognizing this, the work in [3] proposed a deep neural network (DNN) framework to recover a deblurred image from a DP image pair using ground-truth data captured from a Canon DSLR.

Although our work of [3] (Chapter 5) demonstrates state-of-the-art deblurring results, it is restricted by the requirement for accurate ground truth data, which requires DP images to be captured in succession at different apertures. As well as being labor intensive, the process requires careful control to minimize the exposure and motion differences between captures (e.g., see local misalignment in Figure 6.3). Another significant drawback is that data capture is limited to a *single* commercial camera, the Canon 5D Mark IV, the only device that currently provides access to raw DP data and has a controllable aperture.

While datasets exist for defocus estimation, including CUHK [61], DUT [64], and SYNDOF [68], as well as lightfield datasets for defocus deblurring [66] and depth estimation [85,86], none provide DP image views. Our dataset of [3] (Chapter 5) is currently the only source of ground truth DP data suitable for defocus deblur applications but is limited

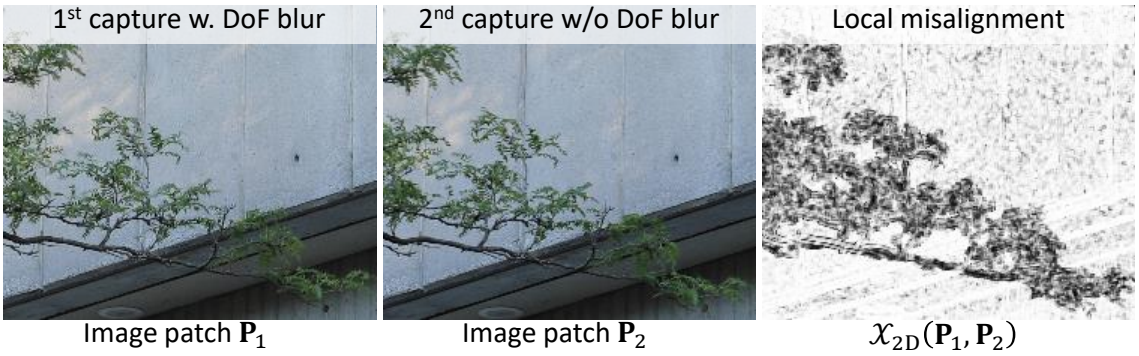


Figure 6.3: Misalignment in the Canon DP dataset [3] due to the physical capture procedure. Patches $\mathbf{P}_1$ and $\mathbf{P}_2$ are cropped from in-focus regions from two captures: the first using a wide-aperture (w/ defocus blur) and the second capture using a narrow-aperture (w/o defocus blur). The 2nd capture is intended to serve as the ground truth for this image pair. The third column shows the 2D cross correlation between the patches $\mathcal{X}_{2D}(\mathbf{P}_1, \mathbf{P}_2)$, which reveals the local misalignment that occurs in such data capture.

to a single device. This lack of data is a severe limitation to continued research on data-driven DP-based defocus deblurring, in particular to applications where the collection of ground truth data is not possible (e.g., fixed aperture smartphones).

Contributions This chapter aims to overcome the challenges in gathering ground-truth DP data for data-driven defocus deblurring. In particular, we propose a generalized model of the DP image acquisition process that allows realistic generation of synthetic DP data using standard computer graphics-generated imagery. We demonstrate that we can achieve state-of-the-art defocus deblurring results using synthetic data only (see Figure 6.1), as well as complement real-image data sets through data augmentation. To demonstrate the generality of the model, we explore a new application domain of video defocus deblurring using DP data and propose a recurrent convolutional network (RCN) architecture that scales from single-image deblurring to video deblurring applications. Additionally, our proposed RCN addresses the issue of patch-wise training by incorporating radial distance learning and improves the deblurring results with a novel multi-scale edge loss. Our comprehensive experiments demonstrate the power of our synthetic DP data generation procedure and show that we can achieve the state-of-the-art results quantitatively and qualitatively with a novel network design trained with this data. The contributions made in

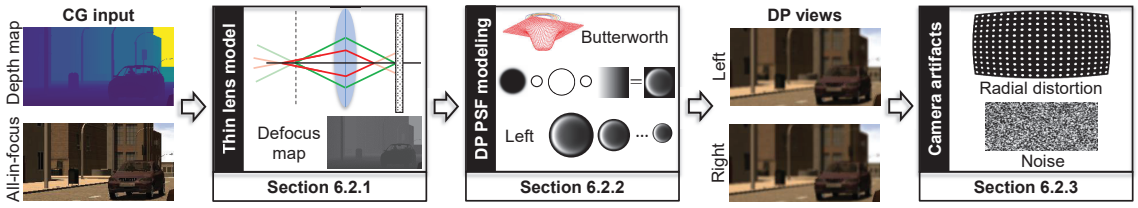


Figure 6.4: An overview of our framework used to synthetically generate dual-pixel (DP) views. Our approach starts with computer-generated (CG) imagery produced with a standard computer graphics package. Starting from this data, we model scene defocus, PSFs related to the DP sensor image formation, and additional artifacts, including radial distortion and sensor noise.

this chapter were published in the International Conference on Computer Vision (ICCV), 2021 [110].

Related links:

- Dataset, code, and trained models: `https://github.com/Abdullah-Abuolaim/recurrent-defocus-deblurring-synth-dual-pixel`
- ICCV presentation: `https://www.youtube.com/watch?v=SxLgE3xwBAQ`

6.2 Modeling Defocus Blur in Dual-Pixel Sensors

Synthetically generating realistic blur has been shown to improve data-driven approaches to both defocus map [68] and depth map estimation [111]. We follow a similar approach in this chapter but tackle the problem of generating realistic defocus blur targeting DP image sensors. For this, we comprehensively model the complete DP image acquisition process with spatially varying PSFs, radial lens distortion, and image noise. Figure 6.4 shows an overview of our DP data generator that enables the generation of realistic DP views from an all-in-focus image and corresponding depth map.

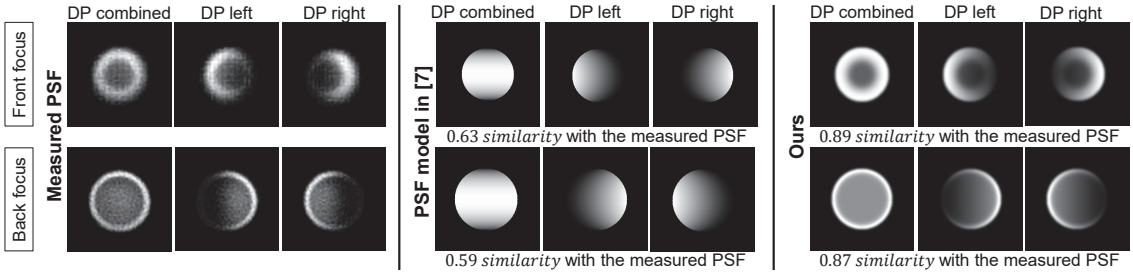


Figure 6.5: Front and back focus DP PSFs. The similarity between two PSFs is measured by the 2D cross correlation. **Left**: Measured DP PSFs from the Canon 5D Mark IV DSLR. **Middle**: DP PSFs as modeled by [4]. **Right**: Our newly proposed model for DP PSFs based on a modified 2D Butterworth filter. Our modeling achieves higher correlation with the real-world measured PSFs.

6.2.1 Thin Lens Model

We model our virtual camera optics using a thin lens model that assumes negligible lens thickness, helping to simplify optical ray tracing calculations [112]. Through this model, we can approximate the circle of confusion (CoC) that represents the PSF for a given point based on its distance from the lens and camera parameters (i.e., focal length, aperture size, and focus distance). This model is illustrated in Figure 6.2, where f is the focal length, s is the focus distance, and F is the f-stop. The distance between the lens and sensor s', and the aperture diameter q are defined as $s' = \frac{f\,s}{s-f}$ and $q = \frac{f}{F}$. Then, the CoC radius r of a scene point P_1 located at distance d from the camera is:

$$r = \frac{q}{2} \times \frac{s'}{s} \times \frac{d-s}{d}. \tag{6.1}$$

6.2.2 Dual-Pixel PSFs

Our recent work in [4] introduced a simplified model for approximating DP PSFs in order to estimate scene depth from a DP views. The approximation in [4] models the PSFs that occurs in the *left* and *right* views of a DP sensor using a nice symmetry property between the *left* and *right* PSFs. This PSF modeling [4] involved a single free parameter only that was directly correlated to the CoC size (Figure 6.5, middle column). Though the model is able to capture the symmetry property observed in a real DP PSF, the overall PSF did not sufficiently reflect the true structure exhibited by real-world PSFs, as illustrated in Figure 6.5's left column. Real DP PSFs exhibit a donut-shaped depletion in the CoC

that is attributed to optical aberrations [72]. More details about of our simplified DP PSF model for depth estimation can be found in Appendix B.

To provide more realistic PSFs for the DP views, we introduce a parametric model based on the 2D Butterworth filter $\mathbf{B}$ [113], defined as follows:

$$\mathbf{B}(x,y) = \left(1 + \left(\frac{D_o}{\sqrt{(x-x_o)^2+(y-y_o)^2}}\right)^{2n}\right)^{-1}, \tag{6.2}$$

where n is the filter order, and D_o is a parameter controlling the 3dB cutoff position. Aiming to capture the donut-shaped structure of the PSF, we define a parametric PSF model based on the Butterworth filter $\mathbf{B}$ as follows:

$$\mathbf{H} = \mathbf{B} \circ \mathbf{C}(x_o, y_o), \tag{6.3}$$

where $\mathbf{C}$ represents a circular disk with radius r matching the CoC radius as calculated in Equation 6.1. The notation $\circ$ denotes the Hadamard product. Both $\mathbf{B}$ and $\mathbf{C}$ are centered at (x_o, y_o). D_o is a function of the radius r and is controlled by the parameter α. The values of $\mathbf{B}$ are re-scaled to $[\beta, 1]$, where the parameter $\beta > 0$ is introduced to control the minimum depletion at the kernel's center (which is always positive based on our observation of PSFs measured from real-world data). With our proposed model, the parameterized PSF $\mathbf{H}$ has a sharp fall-off about the circumference. Therefore, we smooth $\mathbf{H}$ by convolving it with a Gaussian kernel of standard deviation $\kappa \times r$, where $0 < \kappa << 1$.

Our modeling of $\mathbf{H}$ represents the combined DP PSF, which is formed as $\mathbf{H} = \mathbf{H}_l + \mathbf{H}_r$, where $\mathbf{H}_l$ and $\mathbf{H}_r$ are the *left* and *right* DP PSFs, respectively. Similar to the work in [4], we enforce the constraint of horizontal symmetry between $\mathbf{H}_l$ and $\mathbf{H}_r$, and express $\mathbf{H}_r$ as $\mathbf{H}_r = \mathbf{H}_l^f$, where $\mathbf{H}_l^f$ represents the *left* PSF flipped about the vertical axis. $\mathbf{H}_l$ can be shown as $\mathbf{H}$ with a gradual fall-off towards the right direction (see front-focus DP left in Figure 6.5). Mathematically, we denote $\mathbf{H}_l$ as:

$$\mathbf{H}_l = \mathbf{H} \circ \mathbf{M}, \quad \text{s.t.} \quad \mathbf{H}_l \geq \mathbf{0}, \text{ with} \sum \mathbf{H}_l = \tfrac{1}{2}, \tag{6.4}$$

where $\mathbf{M}$ is a 2D ramp mask with a constant decay. This decay can be considered as an intensity fall-off (intensity/pixel) in a given direction. The direction is determined by the sign of the CoC radius calculated based on the thin lens model. The positive sign represents the front focus (i.e., blurring of objects behind the focal plane), whereas the negative sign represents the back focus (i.e., blurring of objects in front of the focal plane).

Figure 6.6: Animated synthetic DP views generated using our framework. **Note: The DP views are animated; click on the image to start the animation. It is recommended to open this PDF in Adobe Acrobat Reader to work properly.**

Our PSF model, parameterized with five parameters, facilitates synthesizing PSF shapes more similar to what we measured in real cameras under different scenarios (see Figure 6.5, right column). From this model, we can generate a bank of representative PSFs based on actual observations from real cameras. Additional details about the calibration procedure, PSF estimation method, and parameter searching are provided in Appendix C.

6.2.3 Modeling Additional Camera Artifacts

Radial lens distortion Radial lens distortion occurs due to lens curvature imperfections causing straight lines in the real world to map to circular arcs in the image plane. This is a well-studied topic with many methods for modelling and correcting the radial distortion (e.g., [114–117]). In our framework, we consider applying radial distortion to the synthetically generated images to mimic this effect found in real cameras. We adopt the widely used division model introduced in [115], as follows:

$$(x_d, y_d) = (x_o, y_o) + \frac{(x_u - x_o, y_u - y_o)}{1 + c_1 R^2 + c_2 R^4 + \cdots}, \tag{6.5}$$

where (x_u, y_u) and (x_d, y_d) are the undistorted and distorted points respectively, and c_i is the i^{th} radial distortion coefficient. R is the radial distance from the image plane center (x_o, y_o). This model enables different types of radial distortion, including barrel and pincushion. We generate representative radial distortion sets at different focal lengths found on cameras. A detailed description of this procedure is provided in Appendix C.

Noise Image noise is the undesirable occurrence of random variations in intensity or color information in images. Our initial input is CG-generated data that is noise-free. In order to synthesize realistic images, we add signal-dependent noise as the last step. We model the noise using a signal-dependent Gaussian distribution where the variance of noise is proportional to image intensity [118, 119]. Let $\mathbf{I}$ be the noiseless image and $\mathbf{N}$ a zero-mean Guassian noise layer; then our modeling of the signal-dependent Gaussian noise is $\mathbf{I}_{\text{noise}} = \mathbf{I} + \mathbf{I} \circ \mathbf{N}$, where $\mathbf{N} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \text{Id})$, and σ controls the noise strength.

6.3 Generating Dual-Pixel Views

In this section we introduce the synthetic dataset used, followed by a description of the procedure to synthetically generate the DP *left* and *right* views. The source of our synthetic example data comes from the street view SYNTHIA dataset [120], which contains image sequences of photo-realistic GC-rendered images from a virtual city. Each sequence has 400 frames on average. The dataset contains a large diversity in scene setups, involving many objects, cities, seasons, weather conditions, day/night time, and so forth. The SYNTHIA dataset also includes the depth-buffer and labelled segmentation maps. In our framework, we use the depth map to apply synthetic defocus blur in the process of generating the DP views.

To blur an image based on the computed CoC radius r, we first decompose the image into discrete layers according to per-pixel depth values, where the maximum number of layers is set to 500. Then, we convolve each layer with our parameterized PSF, blurring both the image and mask of the depth layer. Next, we alpha-blend the blurred layer images in order of back-to-front, using the blurred masks as alpha values. For each all-in-focus video frame $\mathbf{I}_s$, we generate two images – namely, the *left* $\mathbf{I}_l$ and *right* $\mathbf{I}_r$ sub-aperture DP views – as follows (for simplicity, let $\mathbf{I}_s$ be a patch with all pixels from the same depth

Figure 6.7: Animated real DP views from Canon 5D Mark IV DSLR camera. **Note: The DP views are animated; click on the image to start the animation. It is recommended to open this PDF in Adobe Acrobat Reader to work properly.**

layer):

$$\mathbf{I}_l = \mathbf{I}_s * \mathbf{H}_l, \qquad \mathbf{I}_r = \mathbf{I}_s * \mathbf{H}_r, \tag{6.6}$$

where $*$ denotes the convolution operation. Afterwards, the radial distortion is applied on $\mathbf{I}_l$, $\mathbf{I}_r$, and $\mathbf{I}_s$ based on the camera's focal length. Finally, we add signal-dependent noise layers (i.e., $\mathbf{N}_l$ and $\mathbf{N}_r$) for the two DP views that have the same σ, but are drawn independently. The final output defocus blurred image $\mathbf{I}_b$ is equal to $\mathbf{I}_l + \mathbf{I}_r$.

Our synthetically generated DP views exhibit a similar focus disparity to what we find in real data, where the in-focus regions show no disparity and the out-of-focus regions have defocus disparity. We provide animated examples that alternate between the *left* and *right* DP views to assist in visual comparisons between synthetic (Figure 6.6) and real data captured by: Canon 5D Mark IV DSLR camera (Figure 6.7) and Pixel 4 smartphone camera (Figure 6.8). These figures demonstrate that the direction difference between the DP views in both the synthetic and real DP data is the same, where the views are rotating around the focal plane.

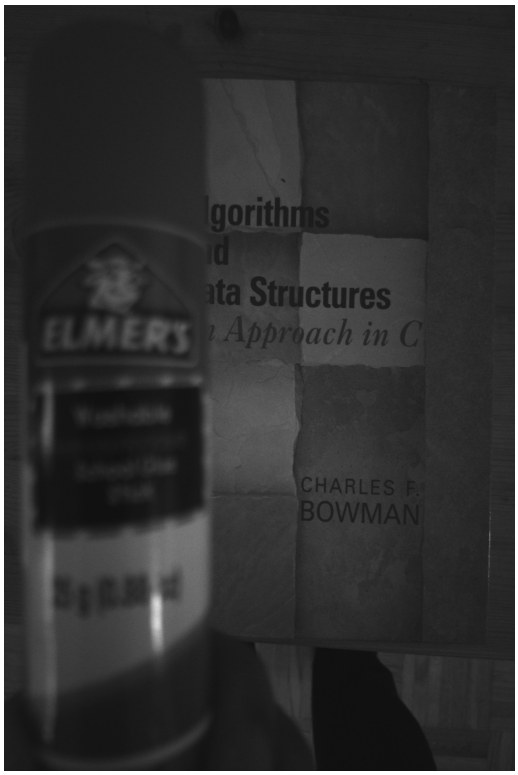


Figure 6.8: Animated real DP views from Pixel 4 smartphone camera. The DP data currently available from the Pixel smartphones are not full-frame, but are limited to only one of the green channels in the raw-Bayer frame. **Note: The DP views are animated; click on the image to start the animation. It is recommended to open this PDF in Adobe Acrobat Reader to work properly.**

6.4 Defocus Deblurring Image Sequences

With the ability to generate synthetic DP data, we can shift our attention to training new RCN-based architectures addressing image sequences (e.g., video) captured with DP sensors. This is possible only by using our synthetic DP data, as no current device allows video DP data capture. As we will show, our method can be used for both image sequences and single-image inputs. In the context of image sequences, the amount of defocus blur changes based on the motion of the camera and scene's objects over time. In the presence of such motion, sample depth variation over a sequence of frames provides useful information for deblurring. Our work is the first to explore the domain of defocus deblurring on image sequences (e.g., video).

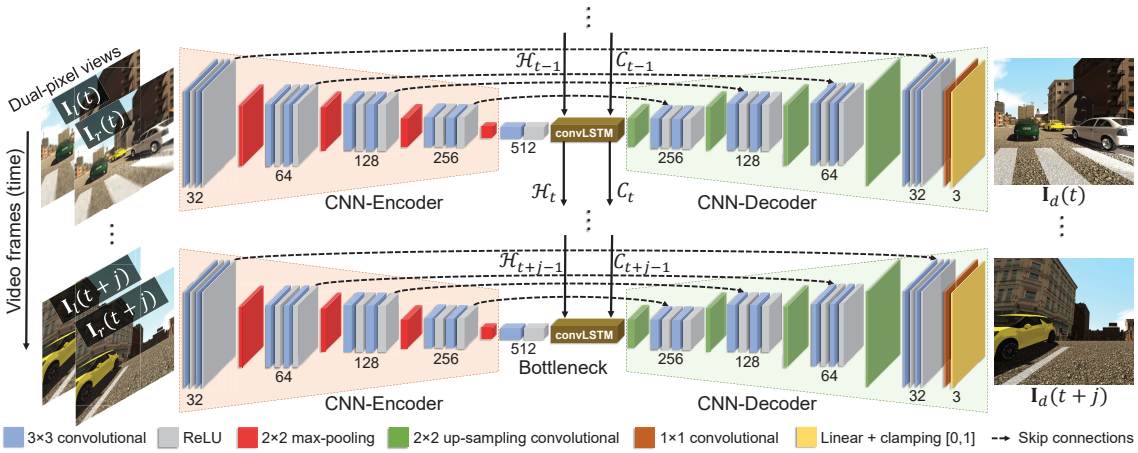


Figure 6.9: Our recurrent dual-pixel deblurring (RDPD) architecture. Our model takes a blurred image sequence, where each image at time t is fed as *left* $\mathbf{I}_l(t)$ and *right* $\mathbf{I}_r(t)$ DP views. The DP views are encoded at the encoder part to feed the convolutional long short term memory (convLSTM) that outputs the hidden state $\mathcal{H}_t$ and memory cell $\mathcal{C}_t$ to the next time point. The convLSTM unit also outputs a feature map o_t that is processed through the decoder part to give the deblurred sharp image $\mathbf{I}_d(t)$. Note: the number of output filters is shown under each convolution operation.

We adopt a data-driven approach for correcting defocus blur. We leverage a symmetric encoder-decoder CNN-based architecture with skip connections between corresponding feature maps [95,96]. Skip connections are widely used in encoder-decoder CNNs and have been found to be effective for image deblurring tasks [3,16]. Our proposed network is also coupled with convLSTM units [2,121,122] to better learn temporal dependencies between multiple frames and to allow variable sequence size. With convLSTM units, the same network remains fully convolutional and can successfully deblur a single image or a sequence of images of arbitrary number. Figure 6.9 shows a detailed overview of our proposed CNN-convLSTM architecture, which we refer to as *recurrent dual-pixel deblurring* (RDPD).

Our architecture is similar to the one in [3], but with the following modifications: (1) convLSTM units are added to the network bottleneck, (2) we train the network using the radial distance patch to address the patch-wise training issue, (3) we introduce a multi-scale edge loss function that helps in recovering sharp edges, (4) the number of nodes are reduced to half at each block to make the model lighter, and (5) the last layer is replaced by a linear layer with a [0,1] clamping as it is found to be more effective in [16].

Table 6.1: Results on the Canon DP dataset from [3]. DPDNet is the pre-trained model on Canon data provided by [3]. DPDNet+ and our RDPD+ are trained with Canon and our synthetically generated DP data. Bold numbers are the best and highlighted in green. The second-best performing results are highlighted in yellow. The testing set consists of 37 indoor and 39 outdoor scenes.

Method	**Indoor**			**Outdoor**			**Indoor & Outdoor**			**Time ↓**
	PSNR ↑	SSIM ↑	MAE ↓	PSNR ↑	SSIM ↑	MAE ↓	PSNR ↑	SSIM ↑	MAE ↓	
EBDB [67]	25.77	0.772	0.040	21.25	0.599	0.058	23.45	0.683	0.049	929.7
DMENet+ [68]	25.70	0.789	0.036	21.51	0.655	0.061	23.55	0.720	0.049	613.7
JNB [70]	26.73	0.828	0.031	21.10	0.608	0.064	23.84	0.715	0.048	843.1
DPDNet [3]	27.48	0.849	0.029	**22.90**	**0.726**	**0.052**	25.13	**0.786**	0.041	0.5
DPDNet+ [3]	27.65	**0.852**	0.028	22.72	0.719	0.054	25.12	0.784	0.042	0.5
Our RDPD+	**28.10**	0.843	**0.027**	22.82	0.704	0.053	**25.39**	0.772	**0.040**	**0.3**

RDPD architecture Given an input video of j consecutive frames that have defocus blur $\{\{\mathbf{I}_l(t), \mathbf{I}_r(t)\}, \cdots, \{\mathbf{I}_l(t+j), \mathbf{I}_r(t+j)\}\}$ (such that $\mathbf{I}_l(t)$ and $\mathbf{I}_r(t)$ are the DP views of the given frame at time t), we first obtain a sequence of compact convolutional features $\{X(t), \cdots, X(t+j)\}$ encoded at the CNN bottleneck – namely, $X(t) = \text{CNN-Encoder}(\mathbf{I}_l(t), \mathbf{I}_r(t))$. Then, the features are fed to a convLSTM as shown in Figure 6.9. We utilize the convLSTM to learn of the temporal dynamics of the sequential inputs. This is achieved by incorporating memory units with the gated operations. The convLSTM also preserves spatial information by replacing dot products with convolutional operations, which is essential for making spatially variant estimation align with the spatially varying DP PSFs. We choose LSTM over RNN because standard RNNs are known to have difficulty in learning long-time dependencies [54], whereas LSTMs have shown the capability to learn long- and short-time dependencies [45].

For the input feature $X(t)$ at time t, our convLSTM leverages three convolution gates – input i_t, output o_t, and forget $\mathcal{F}_t$ – in order to control the signal flow within the cell. The convLSTM outputs a hidden state $\mathcal{H}_t$ and maintains a memory cell $\mathcal{C}_t$ for controlling the state update and output:

$$i_t = \Sigma(W_i^X * X_t + W_i^{\mathcal{H}} * \mathcal{H}_{t-1} + W_i^{\mathcal{C}} \circ \mathcal{C}_{t-1} + b_i), \tag{6.7}$$

$$\mathcal{F}_t = \Sigma(W_{\mathcal{F}}^X * X_t + W_{\mathcal{F}}^{\mathcal{H}} * \mathcal{H}_{t-1} + W_{\mathcal{F}}^{\mathcal{C}} \circ \mathcal{C}_{t-1} + b_{\mathcal{F}}), \tag{6.8}$$

$$o_t = \Sigma(W_o^X * X_t + W_o^{\mathcal{H}} * \mathcal{H}_{t-1} + W_o^{\mathcal{C}} \circ \mathcal{C}_{t-1} + b_o), \tag{6.9}$$

$$\mathcal{C}_t = \mathcal{F}_t \circ \mathcal{C}_{t-1} + i_t \circ \ \tau(W_{\mathcal{C}}^{X} * X_t + W_{\mathcal{C}}^{\mathcal{H}} * \mathcal{H}_{t-1} + b_{\mathcal{C}}), \tag{6.10}$$

$$\mathcal{H}_t = o_t \circ \tau(\mathcal{C}_t), \tag{6.11}$$

where the W terms denote the different weight matrices, and the b terms represent the different bias vectors. Σ and τ are the activation functions of logistic sigmoid and hyperbolic tangent, respectively. Afterwords, the output deblurred image $\mathbf{I}_d$ is obtained by decoding o_t through the decoder part of our encoder-decoder CNN as follows:

$$\mathbf{I}_d(t) = \text{CNN-Decoder}(o_t). \tag{6.12}$$

Radial distance patch Radial distortion and lens aberration make the PSFs vary in radial directions away from the image center. Similar to [3, 69], we perform patch-wise training to avoid the redundancies of full image training and ensure that the input has enough variance. However, this approach breaks the spatial correlation between the image patches as they are fed independently with no knowledge of their location on the image plane. As a result, in addition to the six-channel RGB DP views, we include a single-channel patch that represents the relative radial distance.

Multi-scale edge loss In addition to the MSE loss, we introduce a multi-scale edge loss using a Sobel gradient to guide the network to encourage sharper edges. Our new loss is similar in principle to the single-scale (i.e., 3×3) Sobel loss used in [123], but we modified this loss in two ways: first, we added multiple scales of the Sobel operator (i.e., kernel sizes) in order to capture different edge sizes. Second, we minimized for the horizontal and vertical directions separately, to concentrate more on the direction that is perpendicular to the imaging sensor orientation. For our multi-scale modified edge loss, the vertical G^x and horizontal G^y derivative approximations of the deblurred output I_d and its ground truth I_s are:

$$G_d^x = \mathrm{I}_d * S_{m\times m}^x, \qquad G_d^y = \mathrm{I}_d * S_{m\times m}^y, \tag{6.13}$$

$$G_s^x = \mathrm{I}_s * S_{m\times m}^x, \qquad G_s^y = \mathrm{I}_s * S_{m\times m}^y, \tag{6.14}$$

where $S_{m\times m}^x$ and $S_{m\times m}^y$ are the vertical and horizontal Sobel operators of size m, respectively. The derivative operations are performed at multiple filter sizes. Our new edge loss $\mathcal{L}_{\text{edge}}$ is the mean of multiple scales for each direction x/y and denoted as:

$$\mathcal{L}_{\text{edge}}^{\{x,y\}} = \mathbb{E}[\text{MSE}(G_s^{\{x,y\}}, G_d^{\{x,y\}})]. \tag{6.15}$$

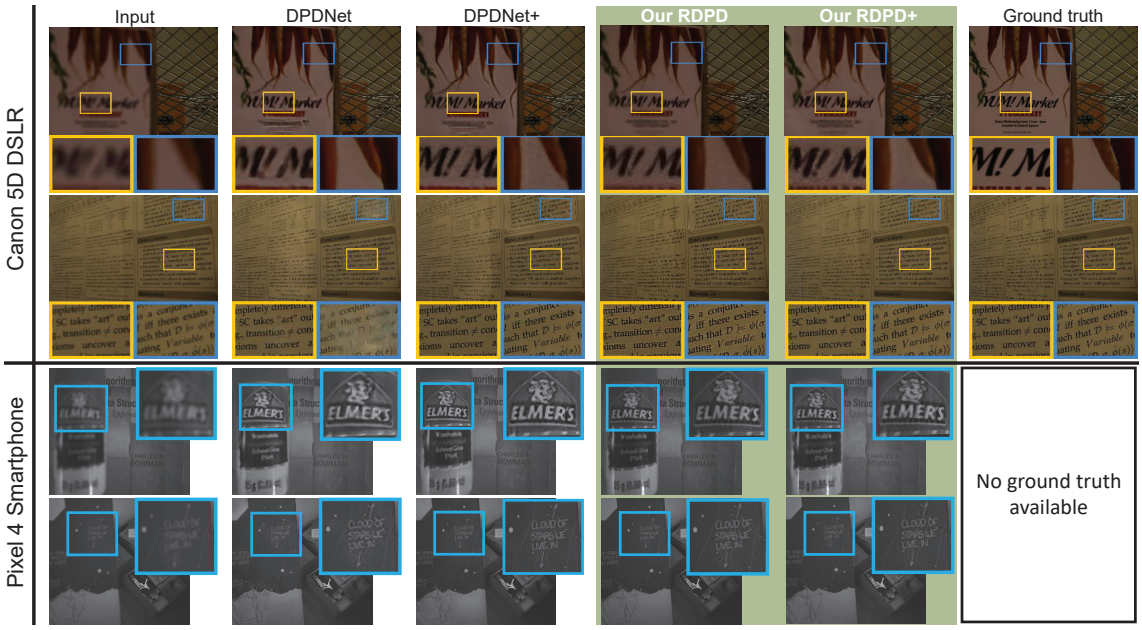


Figure 6.10: Qualitative results. DPDNet [3] is trained on Canon DP data. RDPD is our method trained on synthetically generated DP data only. DPDNet+ and RDPD+ are trained on *both* Canon and synthetic DP data. In general, RDPD and RDPD+ are able to recover more image details. Interestingly, RDPD trained on synthetic data generalizes well to real data from the two tested cameras. Note that there is no ground truth sharp image for Pixel 4, due to the fact smartphones have fixed aperture and thus a narrow-aperture image cannot be captured to serve as a ground truth image. Additionally, we note that the DP data currently available from the Pixel smartphones are not full-frame, but are limited to only one of the green channels in the raw-Bayer frame.

Then the final loss function $\mathcal{L}$ is:

$$\mathcal{L} = \lambda \mathcal{L}_{\text{MSE}} + \lambda_x \mathcal{L}^x_{\text{edge}} + \lambda_y \mathcal{L}^y_{\text{edge}}, \tag{6.16}$$

such that $\mathcal{L}_{\text{MSE}}$ is the typical MSE loss between the output estimated I_d and its ground truth I_s. The λ terms are added to control our final loss.

6.5 Experiments

We evaluate our proposed RDPD and other existing defocus deblurring methods: the DP deblurring network (DPDNet) [3], the edge-based defocus blur (EBDB) [67], the defocus

map estimation network (DMENet) [68], and the just noticeable blur (JNB) [70] estimation. DPDNet [3] is the only method that utilizes DP data for deblurring, and the others [67, 68, 70] use only a single image as input (i.e., I_l) and estimate the defocus map in order to feed it to an off-the-shelf deconvolution method (i.e., [73, 74]). EBDB [67] and JNB [70] are not learning-based methods; thus, we can test them directly. For the learning-based DMENet method, we cannot retrain it with the Canon data [3], as it does not provide the ground truth defocus map. However, with our data generator we are able to generate defocus maps, which allows us to retrain DMENet with our synthetically generated data.

Settings to generate DP data For our DP data generator, we define five camera parameter sets – namely, $\{4, 5, 6\}, \{5, 8, 6\}, \{7, 5, 8\}, \{10, 13, 12\}, \{22, 10, 30\}$ – such that each set represents focal length, aperture size, and focus distance. Given the depth range found in the SYNTHIA dataset [120], these camera sets cover a wide range of front- as well as back-focus CoC sizes. For each image sequence in the SYNTHIA dataset, we generate five sequences based on the predefined camera sets. The radial distortion coefficients are set accordingly for each camera set. For the DP PSFs, we generate many representative PSF shapes by varying the parameters in the given ranges $n \in \{3, 6, 9\}$, $\alpha \in \{0.4, 0.6, 0.8, 1\}$, $\beta \in \{0.1, 0.2, 0.3, 0.4\}$, and $\kappa = 0.14$. Image noise layer strength is chosen randomly, where $\sigma \in \{5e^{-2}, 5.5e^{-2}, \cdots, 5e^{-1}\}$. These parameters are set empirically to model real camera hardware.

We divide the SYNTHIA dataset [120] into training and testing sequences. We generate five sets of blurred images for each image sequence. In total, we synthesize 2023 training and 201 testing blurred DP views. Though our synthetic DP data generator enables an unlimited number of images to be generated, we found this number of images sufficient for training. In addition to our synthetically generated DP data, we use the DP ground truth data from [3] with 300 training, 74 validation, and 76 testing pairs of blurred images (with DP views) and corresponding sharp images.

RDPD settings and training procedure We set the size of the convLSTM to 512 units. For patch-wise training, we fix the size of input and output layers to $512 \times 512 \times 7$ and $512 \times 512 \times 3$, respectively. We initialize the weights of the convolutional layers using He's initialization [100] and use the Adam optimizer [90] to train the model. The initial

learning rate is 5×10^{-5}, which is decreased by half every 40 epochs.

For domain generalization from synthetic to real data [124, 125], we train our model iteratively using mini-batches of real (i.e., single image) as well as synthetic data (i.e., image sequence), where the patches are randomly cropped at each iteration. This type of iterative image/image sequence training becomes feasible since our recurrent model RDPD allows training and testing with any number of frames, and it does not need to be preset beforehand. We set the mini-batch size for the real data iteration to eight batches, because the dataset of real data has only single-image examples (i.e., no image sequences). For the synthetic data iteration, we set the mini-batch size to two sequences each of size four frames. We define three scales for our edge loss – namely, $m \in \{3, 7, 11\}$. The λ terms are found to be effective at $\lambda = 0.5$, $\lambda_x = 0.3$, and $\lambda_y = 0.2$.

To avoid overfitting, the dropout layer in the convLSTM is set to 0.4. Our model converges after 140 epochs. Although we train on image patches, our RDPD (with convLSTM) is fully convolutional and enables testing on full-resolution inputs. To demonstrate the effectiveness of each component in our model, an ablation study of different training settings is provided in Section 6.6.

Single image results We evaluate our proposed RDPD against existing defocus deblurring methods for single-image inputs. For methods that utilize DP views for the input image (i.e., RDPD and DPDNet [3]), we introduce variations on the training data used for more comprehensive evaluations. The variations are RDPD+ and DPDNet+ that are trained on both Canon DP data from [3] combined with synthetic DP data generated by the process described in Section 6.2. The RDPD without the + sign is our baseline trained with synthetically generated DP data only. The DPDNet without the + is trained on Canon data only.

In Table 6.1, we report quantitative results on real Canon DP data from [3] using standard metrics – namely, MAE, PSNR, SSIM, and time. In general, our RDPD+ has the best overall PSNR compared to other methods. Particularly, RDPD+ achieves the best PSNR and MAE for both indoor and combined categories, and all with our lighter-weight network that enabled the fastest inference time.

For the Outdoor dataset, the PSNR of RDPD+ is slightly lower (i.e., 0.08dB) due to the fact that the Outdoor dataset is imperfect as a result of the capturing process (see Figure 6.3). DP cameras do not enable simultaneous capture of the DP images and

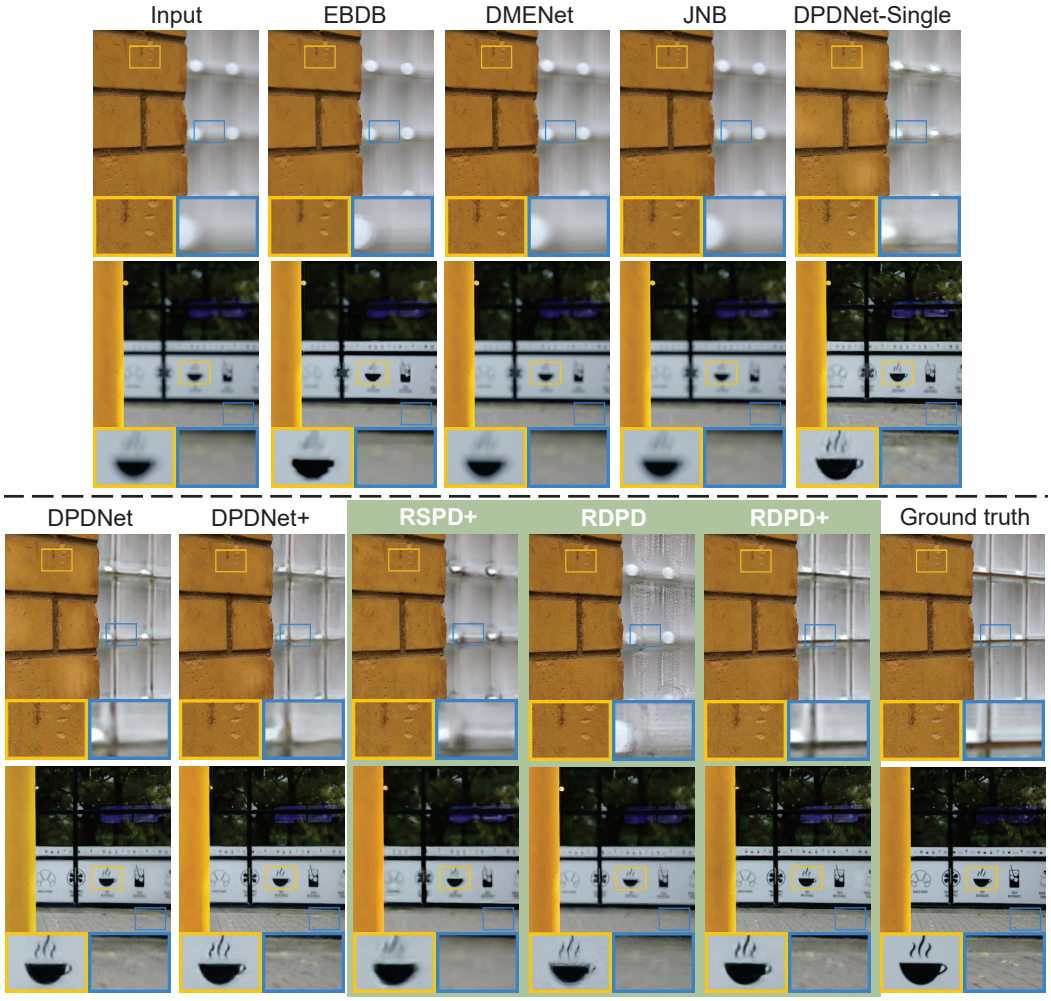


Figure 6.11: Qualitative results on images from Canon dataset [3]. DPDNet [3] is trained on Canon DP data. RDPD is our method trained on synthetically generated DP data only. DPDNet+ and RDPD+ are trained on both Canon and synthetic DP data. DPDNet-Single and RSPD+ are trained on a single DP view (i.e., $\mathbf{I}_l$). In general, RDPD and RDPD+ are able to recover more image details. Interestingly, RDPD trained on synthetic data generalizes well to real data from Canon camera.

corresponding ground truth sharp image (i.e., the image pairs can be captured only in succession at different times). As a result, the outdoor ground-truth is imperfect with small local motion and illumination variations. The Indoor ground-truth is captured in more controlled conditions and has fewer imperfections. The slightly better performance

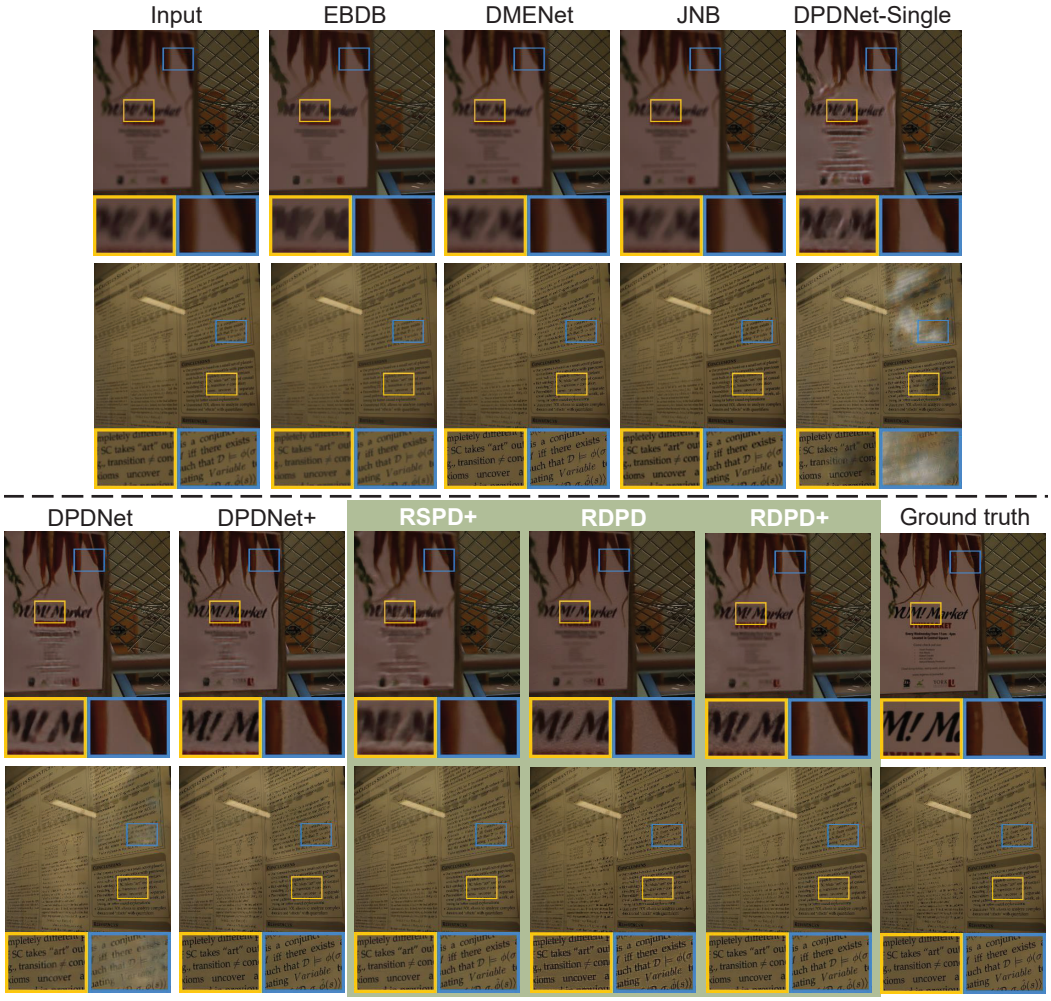


Figure 6.12: Qualitative results on images from Canon dataset [3]. DPDNet [3] is trained on Canon DP data. RDPD is our method trained on synthetically generated DP data only. DPDNet+ and RDPD+ are trained on both Canon and synthetic DP data. DPDNet-Single and RSPD+ are trained on a single DP view (i.e., $\mathbf{I}_l$). In general, RDPD and RDPD+ are able to recover more image details. Interestingly, RDPD trained on synthetic data generalizes well to real data from Canon camera.

of DPDNet for outdoor scenes is because DPDNet is learning to compensate for imperfections in the Outdoor dataset. A key strength of our work is the ability to synthetically generate DP data that is not impeded by imperfections of manual capture. RDPD+ is trained on the Outdoor dataset as well as the synthetically generated data (without such

Table 6.2: Results on our synthetically generated DP data. sRDPD+ is a variation that is trained with single-frame data (**green**=best, **yellow**=second best). Our RDPD+, trained with image sequences, achieves the best results.

Method	PSNR ↑	SSIM ↑	MAE ↓
DPDNet [3]	26.38	0.782	0.034
DPDNet+ [3]	29.84	0.828	0.025
sRDPD+	30.26	0.849	0.020
RDPD+	**31.09**	**0.861**	**0.016**

imperfections) debiasing the result from the imperfect ground-truth. The consequence is reduced fidelity to the imperfect ground-truth (in terms of PSNR/SSIM), but better defocus deblur performance overall. Similar behavior is observed when DPDNet is trained with Canon data and our synthetically generated data (i.e., DPDNet+).

In Figure 6.10, we also provide qualitative results of RDPD compared to other methods on data captured by Canon DSLR and Pixel 4 cameras. In general, RDPD+ is able to recover more details from the input deblurred image. Additionally, Figure 6.10 demonstrates that the baseline RDPD achieves good deblurring results on Canon and Pixel 4 data despite being trained with synthetic data only. This result demonstrates the accuracy of the proposed framework for synthetic DP data generation and the ability of the recurrent model to generalize to different cameras. It can also be seen that DPDNet+ has improved results compared to DPDNet, demonstrating the benefit gained by DPDNet+ through the addition of synthetic DP data on training.

Figure 6.11, Figure 6.12, and Figure 6.13 show more qualitative results on images from the Canon dataset [3]. Figure 6.14 shows more results on images from a Pixel smartphone.

Image sequence results Our RDPD is designed to handle input image sequences. Here, we investigate the improvement gained by training with image sequences vs. single frames. For this comparison, we introduce the RDPD+ variant sRDPD+, which is trained with single-frame inputs.

As previously mentioned, there is no camera that enables access to DP views for video data. Nevertheless, we mimic the same capturing procedure in [3] in order to capture a sequence of images. We performed four captures of the same scene with small camera motion introduced between the captures. Each image has its own DP views and is captured

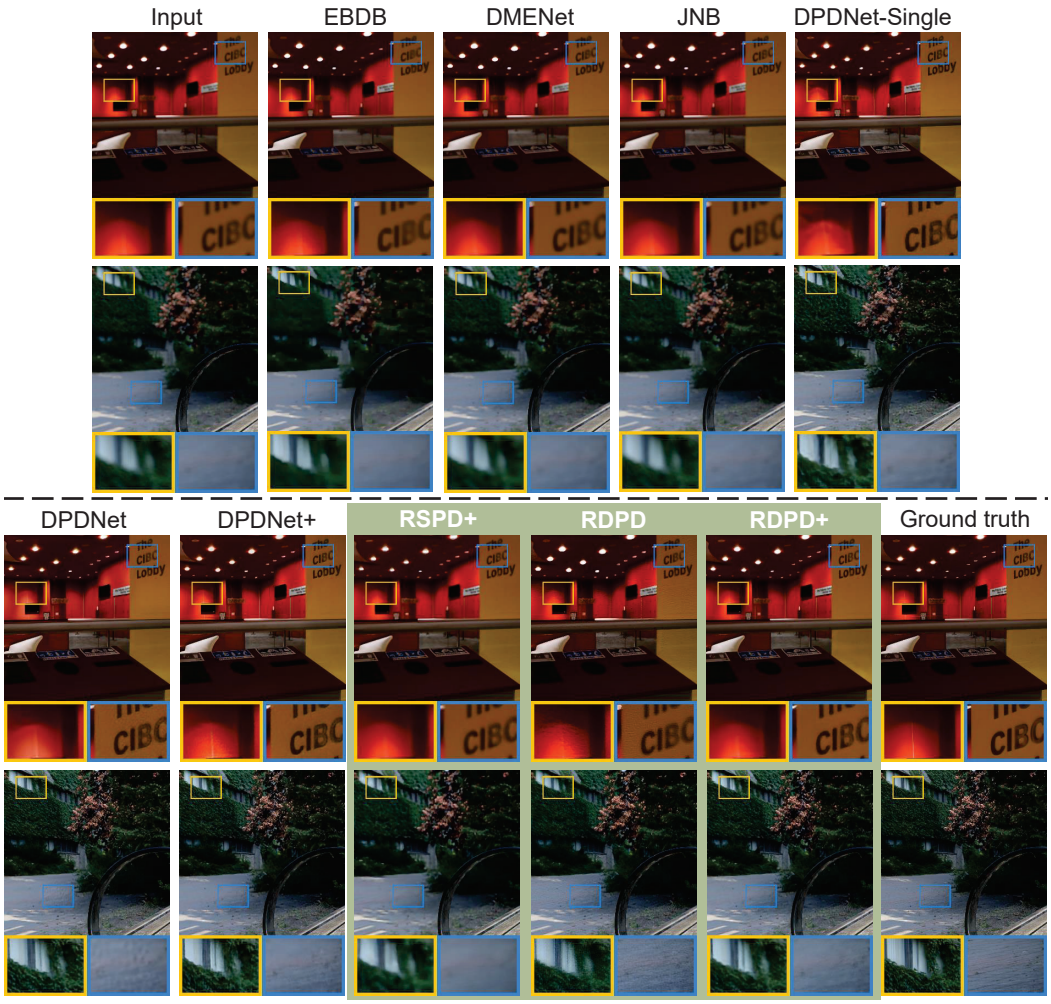


Figure 6.13: Qualitative results on images from Canon dataset [3]. DPDNet [3] is trained on Canon DP data. RDPD is our method trained on synthetically generated DP data only. DPDNet+ and RDPD+ are trained on both Canon and synthetic DP data. DPDNet-Single and RSPD+ are trained on a single DP view (i.e., $\mathbf{I}_l$). In general, RDPD and RDPD+ are able to recover more image details. Interestingly, RDPD trained on synthetic data generalizes well to real data from Canon camera.

at narrow and wide apertures. Figure 6.15 presents the results on the sequence of images. The effectiveness of training with image sequences with RDPD+ can be seen from the average PSNR gain (i.e., +0.4dB) compared to sRDPD+ trained using single-image inputs.

Table 6.2 shows the quantitative results on our synthetically generated DP image

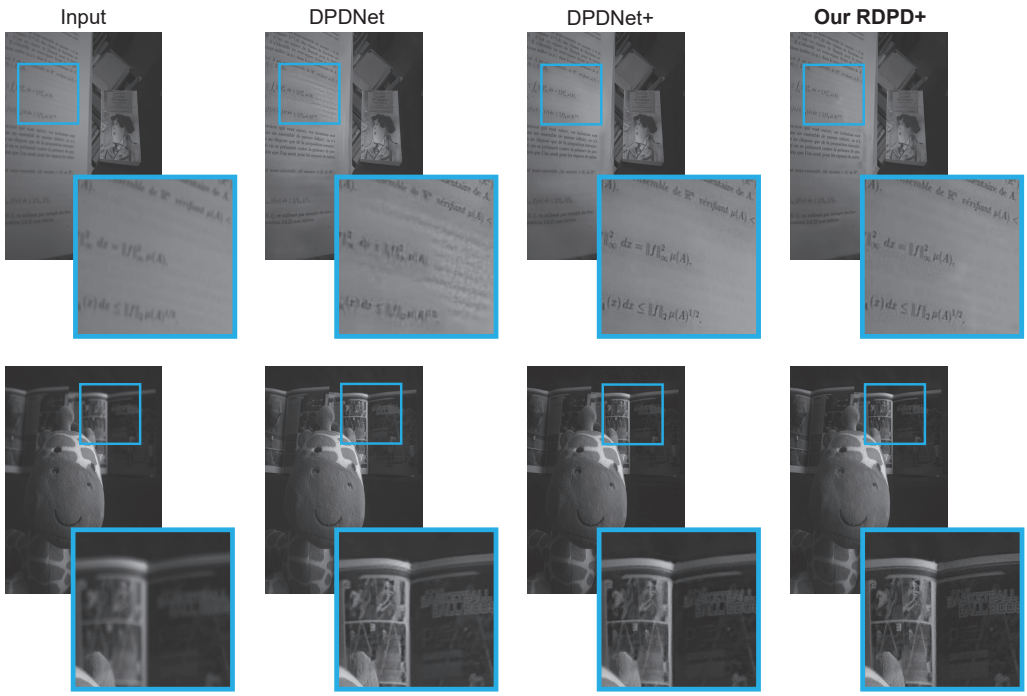


Figure 6.14: Qualitative results on images captured by Pixel smartphone. DPDNet [3] is trained on Canon DP data only. DPDNet+ and RDPD+ are trained on both Canon and synthetic DP data. In general, RDPD+ is able to recover more image details. Interestingly, DPDNet+ achieves better results when it is trained with our synthetic data augmented. Note that there is no ground truth sharp image for Pixel smartphone because smartphones have a fixed aperture. As a result, a narrow-aperture image cannot be captured to serve as a ground truth image. Additionally, we note that the DP data currently available from the Pixel smartphones are not full-frame but are limited to only one of the green channels in the raw-Bayer frame

sequences. Our method RDPD+ (trained on multiple frames) achieves the best results as it utilizes the convLSTM architecture to better model the temporal dependencies in an image sequence. Recall that our RDPD network is lighter and has a much lower number of weights compared to DPDNet.

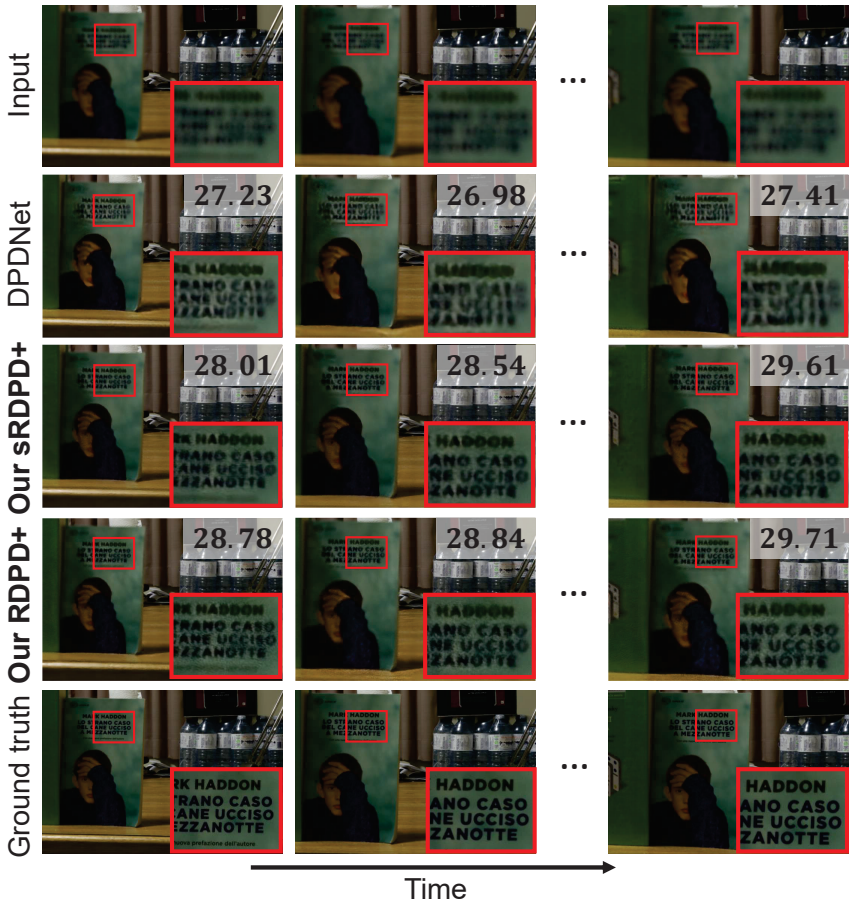


Figure 6.15: Results on a Canon 5D DSLR image sequence. PSNR is shown for each deblurred image. sRDPD+ has a 0.4dB lower PSNR on average when it is trained with a single frame compared to our multi-frame method – that is, RDPD+.

6.6 Ablation Study

This section investigates the usefulness of our synthetically generated DP data along with our novel recurrent dual-pixel deblurring architecture (RDPD). To this aim, we divide the ablation study into two parts:

- Explore the effectiveness of our DP data generator components (Section 6.6.1), including: (1) our parametric DP-PSF model vs. the DP-PSF model presented in [4], (2) radially distorted DP data vs. undistorted ones, and (3) training with dual views vs. training with a single DP view.

Table 6.3: Results on indoor and outdoor scenes combined from the Canon DP dataset [3]. Bold numbers are the best. RDPD+ (PSF [4]) is a variation trained on DP data generated using the PSF model from [4]. Our RDPD+, trained on DP data generated using our parametric DP-PSF, demonstrates +0.7dB higher PSNR and reflects the power of our realistic PSF modeling.

Variation	PSNR ↑	SSIM ↑	MAE ↓
RDPD+ (PSF [4])	24.69	0.752	0.044
RDPD+ (our PSF)	**25.39**	**0.772**	**0.040**

- Investigate the effectiveness of each component added to our RDPD model (Section 6.6.2), including: (1) the utility of adding the radial patch distance mask to the input and (2) our new edge loss vs. traditional Sobel loss.

Note that all subsequent experiments are conducted with variations of RDPD+. Each variation represents a single change, where the rest remains similar to RDPD+ as described in Section 6.4. All the variations are tested on Canon DP data from [3], since it is the only data that we can have access to real ground truth DP data and thus enables us to report quantitative results.

6.6.1 Utility of DP Data Generator Components

Our parametric DP-PSF While our parametric DP-PSF model already has a higher correlation with the estimated PSFs from real cameras, we further investigate the effect on RDPD+ when trained with data generated using other DP-PSFs. In this study, we compare our RDPD+ that is trained with DP data generated using our parametric DP-PSF model against the DP data generated using the DP-PSF model in [4].

The quantitative results reported in Table 6.3 demonstrates the power of training RDPD+ with DP data that is generated using our realistically modeled PSFs, where there is an increase in PSNR of +0.7dB.

Radial distortion For more realistic modeling, we considered applying radial distortion in the proposed DP data generator. In this study, we examine the proposed deblurring RDPD+ model's behavior when it is also trained with data that is not radially distorted.

Table 6.4: Results on indoor and outdoor scenes combined from the Canon DP dataset [3]. Bold numbers are the best. RDPD+ trained with radially distorted DP data achieves +0.2dB higher PSNR when tested on real images, in which applying radial distortion on the synthetically generated DP data helped RDPD+ to learn the spatially varying PSFs shapes found in real cameras.

Variation	PSNR ↑	SSIM ↑	MAE ↓
RDPD+ (w/o distortion)	25.19	0.758	0.041
RDPD+ (w/ distortion)	**25.39**	**0.772**	**0.040**

Table 6.5: Results on indoor and outdoor scenes combined from the Canon DP dataset [3]. Bold numbers are the best. RSPD+: recurrent single-pixel deblurring trained with a single DP view (i.e., left view). RDPD+: trained with left and right DP views. Utilizing DP views to train our RDPD+ leads to +1.15dB PSNR gain and is essential for better defocus deblurring.

Variation	PSNR ↑	SSIM ↑	MAE ↓
RSPD+	24.24	0.726	0.045
RDPD+	**25.39**	**0.772**	**0.040**

In Table 6.4, we present the quantitative results of training RDPD+ with data generated with and without radial distortion. The results demonstrate the effectiveness of modeling the radial distortion during synthesizing the DP images, where RDPD+ trained with radially distorted data leads to a +0.2dB PSNR gain.

Dual views vs. single view Following [3], we explore the effect of training RDPD+ with DP views vs. training with a single view (i.e., the traditional approach of single image deblurring [67,68,70]). To this aim, we introduce a recurrent single-pixel deblurring model variant (RSPD+) and compare it with our proposed RDPD+ model. Quantitative results in Table 6.5 demonstrates that training with DP views is crucial in order to perform better defocus deblurring where there is a +1.15dB PSNR gain.

Table 6.6: Results on indoor and outdoor scenes combined from the Canon DP dataset [3]. Bold numbers are the best. RDPD+ has an improved results when it is trained with the radial distance patch.

Variation	PSNR ↑	SSIM ↑	MAE ↓
RDPD+(w/o radial distance)	25.00	0.756	0.042
RDPD+(w/ radial distance)	**25.39**	**0.772**	**0.040**

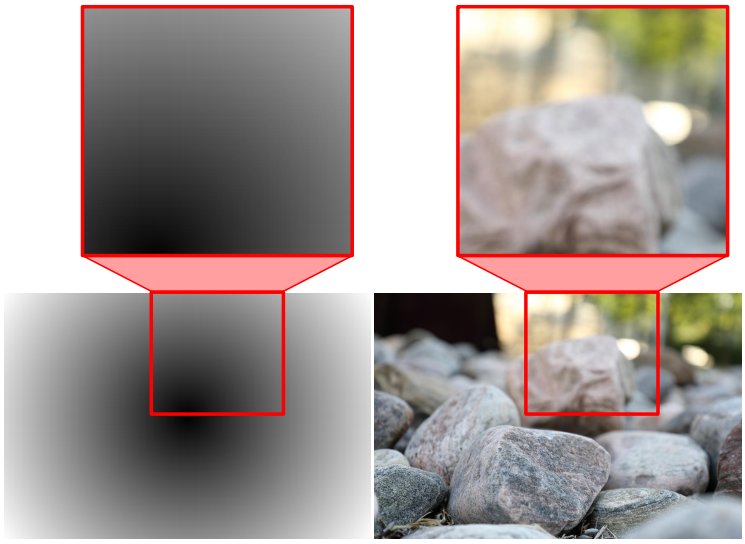


Figure 6.16: The radial distance patch used to assist RDPD+ training and address the issue of patch-wise training.

6.6.2 Effectiveness of RDPD Components

Radial distance patch for patch-wise training We introduced training with the radial distance patch to feed each pixel's spatial location in the cropped patch and address the patch-wise training issue (i.e., the network does not see the full image or the relative position of the patch in the full image). Training in this manner is important as the PSFs are spatially varying in the radial direction away from the image center. Figure 6.16 shows an example of the cropped radial distance patch used to assist our training. To investigate the effectiveness of training with the radial distance patch, we also train RDPD+ without it and report the results in Table 6.6. RDPD+ has a gain in PSNR of +0.4dB when trained with the radial distance patch.

Table 6.7: Results on indoor and outdoor scenes combined from the Canon DP dataset [3]. Bold numbers are the best. **RDPD+(0)**: trained without the edge loss. **RDPD+(1)**: trained with a 3×3 single-scale Sobel loss similar to [123]. **RDPD+(3)**: trained with our three-scale edge loss. RDPD+ trained with our multi-scale edge loss has the best results for all metrics.

Variation	PSNR ↑	SSIM ↑	MAE ↓
RSPD+(0)	25.06	0.765	0.042
RSPD+(1) [123]	25.11	0.763	0.042
RDPD+(3)	**25.39**	**0.772**	**0.040**

Table 6.8: Results on indoor and outdoor scenes combined from the Canon DP dataset [3]. Bold numbers are the best. RDPD+ has about -0.5dB PSNR drop in performance, when both our radial distance patch and multi-scale edge loss are removed from the RDPD+'s training.

Variation	PSNR ↑	SSIM ↑	MAE ↓
RDPD+(w/o both)	24.90	0.754	0.042
RDPD+(w/ both)	**25.39**	**0.772**	**0.040**

Our multi-scale edge loss As mentioned in Section 6.4, we introduced the multi-scale edge loss based on the Sobel gradient operator to recover sharper details at different edge sizes. To examine our edge loss function's effectiveness, we train RDPD+ with different variations of edge loss scales denoted as RDPD+(m), where m is the number of scales used. In particular, we introduce RDPD+(0) (trained without the edge loss), RDPD+(1) (trained with a 3×3 single-scale Sobel loss similar to [123]), and RDPD+(3) (trained with our three-scale edge loss). Table 6.7 shows the quantitative results, in which training with our multi-scale edge loss achieves the best results for all metrics.

Effect of radial distance and edge loss together We also examine the performance when our radial distance patch and multi-scale edge loss are removed from the RDPD+'s training. Table 6.8 shows the results, where there is a PSNR drop of -0.5dB, indicating the usefulness of the additional proposed components.

6.7 Conclusion

We proposed a novel framework to generate realistic DP data by modeling the image formation steps present on cameras with DP sensors. Our framework helps to address the current challenges in capturing DP data. Utilizing our synthetic DP data, we also proposed a new recurrent convolutional architecture that is designed to reduce defocus blur in image sequences. We performed a comprehensive evaluation of existing deblurring methods, and demonstrated that our synthetically generated DP data and recurrent convolutional model achieve state-of-the-art results quantitatively and qualitatively. Furthermore, our proposed framework demonstrates the ability to generalize across different cameras by training on synthetic data only. We believe our DP data generator will help spur additional ideas about defocus deblurring and applications that leverage DP data. Our dataset, code, and trained models are publicly available at: `https://github.com/Abdullah-Abuolaim/recurrent-defocus-deblurring-synth-dual-pixel`

Chapter 7

Single Image Defocus Deblurring

Many camera sensors use a dual-pixel (DP) design that operates as a rudimentary light field providing two sub-aperture views of a scene in a single capture. However, few cameras allow access to DP data at the consumer level. Unlike the previous chapters (i.e., Chapter 5 and Chapter 6), we tackle, in this chapter, the problem of DoF extension in the absence of DP views at inference time.

The DP sensor was developed to improve how cameras perform autofocus. Since the DP sensor's introduction, researchers have found additional uses for the DP data, such as depth estimation, reflection removal, and defocus deblurring. We are interested in the latter task of defocus deblurring. In particular, we propose a single-image deblurring network that incorporates the two sub-aperture views into a multi-task framework. Specifically, we show that jointly learning to predict the two DP views from a single blurry input image improves the network's ability to learn to deblur the image. Our experiments show this multi-task strategy achieves +1dB PSNR improvement over state-of-the-art defocus deblurring methods. In addition, our multi-task framework allows accurate DP-view synthesis (e.g., $\sim$ 39dB PSNR) from the single input image. These high-quality DP views can be used for other DP-based applications, such as reflection removal. As part of this effort, we have captured a new dataset of $7,059$ high-quality images to support our training for the DP-view synthesis task.

(a) Blurred input (b) Result of [67] (c) Result of [68]

(d) Our result (e) Our DP views (f) GT DP views

Figure 7.1: Results from our multi-task framework. (a) Input image with DoF blur. Deblurring results of [67] and [68] are shown in (b) and (c), respectively. (d) Our result. (e) Our reconstructed DP views. (f) Ground-truth DP views. Our multi-task method has better deblurring results and is able to produce accurate DP views from a *single-image* input. **Note: The DP views are animated; click on the image to start the animation. It is recommended to open this PDF in Adobe Acrobat Reader to work properly.**

7.1 Introduction

We are interested in reducing the defocus blur present in captured images. Defocus blur occurs at scene points that are captured outside a camera's depth of field (DoF). Reducing defocus blur is challenging due to the nature of the spatially varying point spread functions (PSFs) that vary with scene depth [71, 72]. Most of the existing DoF blur reduction methods [66–70] approach the problem in two stages: (1) estimate the defocus map of the input and (2) apply off-the-shelf non-blind deconvolution (e.g., [73, 74]) guided by the estimated defocus map. The performance of these methods is bounded by the DoF map estimation and the effectiveness of the non-blind deconvolution. Additionally, due to the two-stage approach, these methods have a long processing time.

Our recent work in [3] (Chapter 5) proposed a DoF extension (i.e., defocus deblurring) method that leveraged the availability of dual-pixel (DP) sensor data. This method trained a deep neural network (DNN) that uses the DP sensor's two sub-aperture views as input to predict a single deblurred image. The effectiveness of our method in [3] (Chapter 5) is attributed to the DNN's ability to learned the amount of spatially varying defocus blur from the two DP views. This idea stems from the way the DP sensors work. DP sensors were developed as a means to improve the camera's AF system. The DP design produces two sub-aperture views of the scene that exhibit differences in phase that are correlated to the amount of defocus blur as shown in Figure 7.1-f (details in Section 7.2). A camera adjusts the lens position to minimize phase differences in the two DP views, resulting in a final in-focus image. Researchers have been quick to leverage the DP sub-images for tasks beyond autofocus [14, 23], including depth map estimation [4, 25, 126], reflection removal [5], and synthetic DoF [27].

One notable drawback of using DP data is that most cameras do not provide easy access to the DP sensor's two sub-aperture views. Although DP sensors are used by many cameras, only two cameras currently provide access to DP images (i.e., Canon 5D DSLR camera [3–5] and Google Pixel series smartphone camera [25, 27, 126]). Even for these devices accessing the DP data has caveats. For example, the Canon 5D requires special software to extract the two views from a saved RAW image and process them to an sRGB color space. The Google Pixel device requires a special binary and provides DP data only for the green channel of the RAW image. These limitations make the use of DP data at inference time impractical. For instance, we investigated the effectiveness of performing

defocus estimation in a linear color space (i.e., CIE XYZ color space) by unprocessing the non-linear sRGB images (more details of are provided in Appendix D).

In the context of defocus deblurring, training data is required in the form of paired images—one sharp and one blurred. Training images are obtained by placing a camera on a tripod and capturing an image using a wide aperture (i.e., blurred image with shallow DoF), followed by a second image captured using a narrow aperture (i.e., target sharp image with large DoF). Care must be taken to ensure that the camera is not moved between aperture adjustments and that the scene content remains stationary. Such data acquisition is a time-consuming process and does not facilitate collecting larger datasets—for instance, our recent DP defocus deblurring dataset [3] contains only 500 of such pairs.

The aforementioned drawbacks of accessing DP data at inference time and the challenges in capturing blurry/sharp paired data for training serve as the impetus for our proposed multi-task learning framework. In particular, our method focuses on conventional single-image input. And, while we cannot remove the need for blurred/sharp training data entirely, we demonstrate that the performance of defocus deblurring is improved by incorporating the joint training of predicting the DP views. The training of the DP-view reconstruction task requires only the capture of unpaired DP images in an unrestricted manner with minimal effort. Because we only need access to DP information at training time, inference time becomes much more practical.

Contributions We introduce a multi-task DNN framework to jointly learn single-image defocus deblurring and DP-based view prediction as shown in Figure 7.1. We show that training a DNN to both deblur the image and predict the two sub-aperture DP views improves deblurring results by up to +1dB PSNR over existing state-of-the-art methods. To facilitate this effort, we have captured an unpaired dataset with varying DoF blur consisting of $2,353$ high-quality full-frame images using a DP camera. This gives a total of $7,059$ images—$2,353$ conventional images and their corresponding two sub-aperture DP views. We also introduce novel loss functions based on DP image formation to help the network avoid ambiguity that arises in DP data. Extensive experiments show our results outperform existing single-image DoF deblurring both quantitatively and qualitatively. While our main goal is defocus deblurring, we conclude by showing how our DNN model has the added advantage of being able to produce high-quality DP views that can be used for tasks such as reflection removal and multi-view synthesis. The work in this chapter was

accepted for publication at the Winter Conference on Applications of Computer Vision (WACV), 2022 [127].

Related links:

- Dataset, code, and trained models: `https://github.com/Abdullah-Abuolaim/multi-task-defocus-deblurring-dual-pixel-nimat`
- The arXiv paper: `https://arxiv.org/pdf/0000.pdf`

7.2 Dual-pixel Image Formation

We start with a brief overview of DP sensors. A DP sensor uses two photodiodes at each pixel location with a microlens placed on the top of each pixel site, as shown in Figure 7.2-a. As previously mentioned, this design was developed to improve camera autofocus by functioning as a simple two-sample light field camera. The two-sample light field provides two sub-aperture views of the scene and, depending on the sensor's orientation, the views can be referred to as left/right or top/down pairs; we follow the convention of prior work [3–5] and refer to them as the left/right pair, denoted as $\mathbf{I}_l$ and $\mathbf{I}_r$. The light rays coming from scene points that are within the camera's DoF exhibit little to no difference in phase between the views. On the other hand, light rays coming from scene points outside the camera's DoF exhibit a noticeable defocus disparity in the *left-right* views. The amount of defocus disparity is correlated to the amount of defocus blur.

Unlike traditional stereo, the difference between the DP views can be modeled as the latent sharp image being blurred in two different directions using a half-circle PSF [4]. This is illustrated in the resultant circle of confusion (CoC) of Figure 7.2-c. On real DP sensors, the ideal case of a half-circle CoC is only an approximation due to constraints of the sensor's construction and lens array. These constraints allow a part of the light ray bundle to leak into the other-half dual pixels (see half CoC of left/right views in Figure 7.2-c). We can describe the DP image formation as follows. Let $\mathbf{I}_s$ be a latent sharp image patch and $\mathbf{H}_l$ and $\mathbf{H}_r$ are the left/right PSFs; then the DP $\mathbf{I}_l$ and $\mathbf{I}_r$ can be represented as:

$$\mathbf{I}_l = \mathbf{I}_s * \mathbf{H}_l, \qquad \mathbf{I}_r = \mathbf{I}_s * \mathbf{H}_r, \tag{7.1}$$

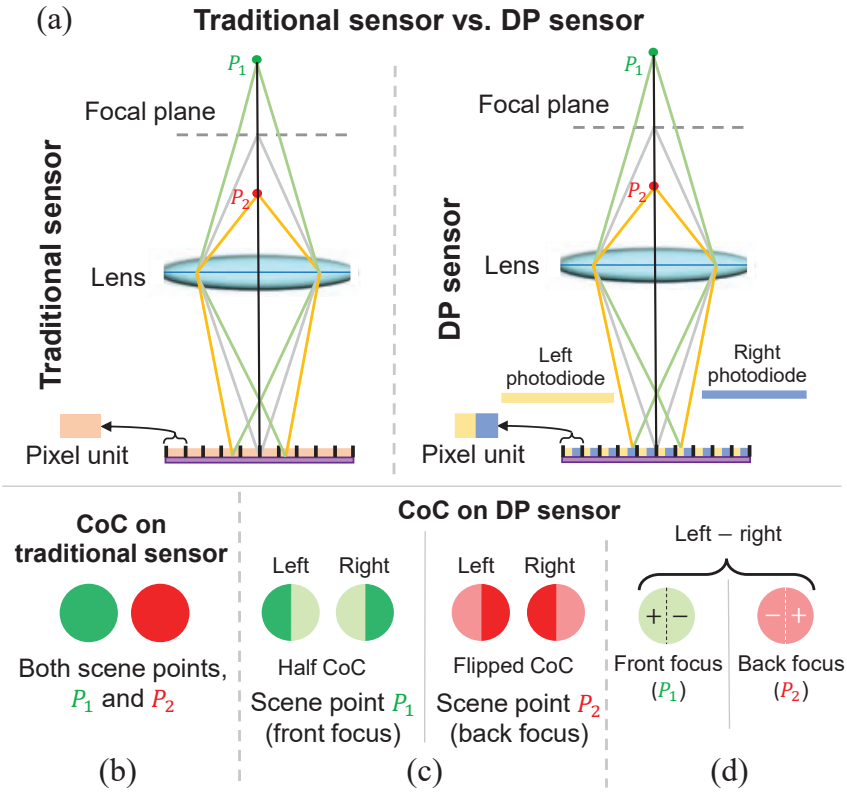


Figure 7.2: DP sensor image formation via DoF circle of confusion (CoC) formation. (a) Traditional sensor vs. DP sensor. (b) and (c) are the CoC formation on the 2D imaging sensor of two scene points, $P1$ and $P2$. On the two DP views, the half-CoC flips direction if the scene point is in front or back of the focal plane. (d) shows the subtracted DP views in the front/back focus cases, where the $+/-$ sign reveals the front/back focus ambiguity.

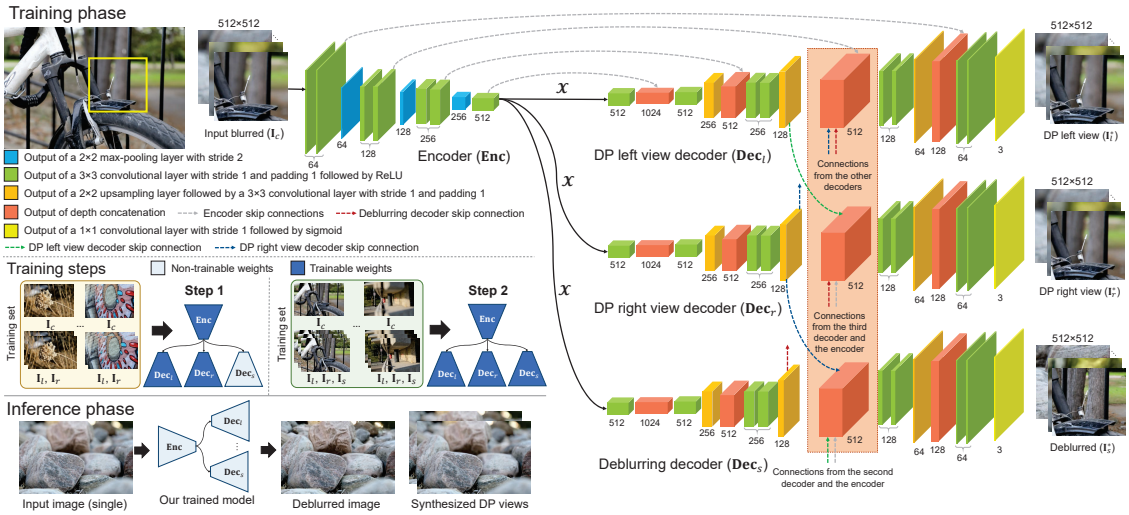


Figure 7.3: An overview of the proposed multi-task learning framework. We adopt a single-encoder multi-decoder DNN. Our multi-task DP network (MDP) takes a single input image ($\mathbf{I}_c$) and outputs three images—namely, left ($\mathbf{I}_l$) and right ($\mathbf{I}_r$) DP views, and a deblurred (sharp) version ($\mathbf{I}_s$). MDP has two stages of weight sharing between the three decoders (i.e., $\mathbf{Dec}_l$, $\mathbf{Dec}_r$, and $\mathbf{Dec}_s$): early at the encoder ($\mathbf{Enc}_l$) latent space $\mathcal{X}$ and middle at the box highlighted in orange. Our MDP is trained in two steps, where the $\mathbf{Dec}_s$ is frozen in the first step and resumed in the next based on the intended task.

$$\mathbf{H}_r = \mathbf{H}_l^f, \tag{7.2}$$

where $*$ denotes the convolution operation and $\mathbf{H}_l^f$ is the flipped $\mathbf{H}_l$. The two views $\mathbf{I}_l$ and $\mathbf{I}_r$ are combined to produce the final image provided by the camera $\mathbf{I}_c$ as follows:

$$\mathbf{I}_c = \mathbf{I}_l + \mathbf{I}_r. \tag{7.3}$$

Another interesting property of the DP PSFs is that the orientation of the "half CoC" of each left/right view reveals if the scene point is in front or back of the focal plane, as shown in the subtracted views of the two scene points, P_1 and P_2 in Figure 7.2-d. These DP properties are useful cues that will be considered when formulating the DP-based loss functions in Section 7.3.2.

7.3 Multi-Task Learning Framework

Multi-task learning has been successfully used for various computer vision tasks [128–130]. In this work, we adopt a multi-task framework in order to leverage the strong connection between defocus deblurring and DP-view synthesis as they require encoding information regarding the defocus blur present at each pixel in the input image. Towards this goal, we propose a single-encoder multi-decoder DNN that takes a single input image and decomposes it into DP left/right views along with the deblurred version. Figure 7.3 provides an overview of our proposed framework.

7.3.1 Network Architecture

We adopt a symmetric single-encoder multi-decoder DNN architecture with skip connections between the corresponding feature maps [95, 96] (see Figure 7.3). We refer to our DNN model as the multi-task DP network (MDP). The three decoder branches have an early-stage weight sharing at the end of the encoder part. We add middle-stage weight sharing as indicated in the orange box of Figure 7.3. Each block in the middle-stage sharing receives two skip connections from the corresponding feature maps from the other two decoders. This type of multi-decoder stitching—that guarantees weight sharing at multiple stages—provides multiple communication layers that can further assist the multi-task joint training. We avoid adding late-stage weight sharing as the sharpness of the deblurred image can be affected by the half-PSF blur present in feature maps of the synthesized DP views at these later stages. The proposed model has a sufficiently large receptive field that is able to cover larger spatially varying defocus PSFs. An ablation study is provided in Section 7.4.3 that validates the multi-decoder stitching design.

With the proposed architecture, the encoder **Enc** task is to map the input image into a latent space $\mathcal{X}$ as follows:

$$\mathcal{X} = \mathbf{Enc}(\mathbf{I}_c). \tag{7.4}$$

This latent space can be viewed as a defocus estimation space in which both tasks share a common goal that requires a notion of the PSF size at each pixel in the input image. This latent space representation $\mathcal{X}$ is then passed to the three decoders—namely, left and right DP-view decoders ($\mathbf{Dec}_l$ and $\mathbf{Dec}_r$) and the defocus deblurring (i.e., sharp image)

decoder ($\mathbf{Dec}_s$)—in order to produce the output estimations as follows:

$$\mathbf{I}_l^* = \mathbf{Dec}_l(\mathcal{X}), \quad \mathbf{I}_r^* = \mathbf{Dec}_r(\mathcal{X}), \quad \mathbf{I}_s^* = \mathbf{Dec}_s(\mathcal{X}). \tag{7.5}$$

7.3.2 DP-Based Loss Function

It is important to consider how the DP images are formed when designing loss functions to ensure the training process for the two DP views satisfies DP properties. We have observed empirically that a traditional mean squared error (MSE) loss, computed between the ground truth (GT) and reconstructed DP views, drives the network to a local minimum, where the difference between the reconstructed DP views is estimated as an explicit shift in the image content. This observation makes the MSE alone not sufficient to capture the flipping property of DP PSFs (i.e., the PSF reverses direction if it is in front of the focal plane—see Figure 7.2-c). Therefore, we introduce a novel DP-loss based on Equation 7.3 that imposes a constraint on the DP-view reconstruction process as follows:

$$\mathcal{L}_C = \frac{1}{n}\sum_n (\mathbf{I}_c - (\mathbf{I}_l^* + \mathbf{I}_r^*))^2, \tag{7.6}$$

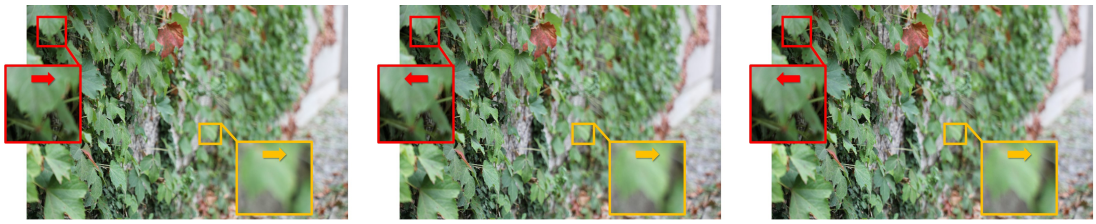
where $\mathbf{I}_c$ is the input combined image and $\mathbf{I}_l^*$ and $\mathbf{I}_r^*$ are the estimated DP views. Our $\mathcal{L}_C$ encourages the network to optimize for the fundamental DP image formation (i.e., Equation 7.3).

While $\mathcal{L}_C$ assists the network to learn that the combined left/right views should sum to the combined image, the front/back focus flipping direction remains ambiguous to the network. To address this ambiguity, we introduce a new view difference loss $\mathcal{L}_D$ to capture the flipping sign direction as follows:

$$\mathcal{L}_D = \frac{1}{n}\sum_n ((\mathbf{I}_l - \mathbf{I}_r) - (\mathbf{I}_l^* - \mathbf{I}_r^*))^2, \tag{7.7}$$

where $\mathbf{I}_l$ and $\mathbf{I}_r$ are the GT DP left and right views, respectively. Figure 7.2-d shows the sign difference in the front/back focus cases when the views are subtracted, which gives a cue for the network to learn the PSF flipping direction when penalizing view difference in the loss—namely, $\mathcal{L}_D$. Figure 7.4 provides a visual analysis of the observations that led to our two DP loss functions—namely, $\mathcal{L}_C$ and $\mathcal{L}_D$. See Section 7.4.3 for an ablation to demonstrate their effectiveness.

(a) Without $\mathcal{L}_C$ loss term (b) With $\mathcal{L}_C$ loss term (c) Ground truth



(d) Without $\mathcal{L}_D$ loss term (e) With $\mathcal{L}_D$ loss term (f) Ground truth

Figure 7.4: An analysis and visual comparison to reflect the effectiveness of our proposed $\mathcal{L}_C$ and $\mathcal{L}_D$ DP-based loss terms. (b) shows that training with $\mathcal{L}_C$ helps the network to capture the flipping kernel (yellow patch) and accurate colors (red patch) compared to the one in (a). (e) demonstrates that training with $\mathcal{L}_D$ can assist the network to learn the flipping direction in the front (yellow patch) and back focus (red patch), where the views rotate around the focal plane as shown in the GT. **Note: The DP views are animated; click on the image to start the animation. It is recommended to open this PDF in Adobe Acrobat Reader to work properly.**

7.3.3 Dual-Pixel Datasets

Our new DP dataset We collected a new DP dataset of $2,353$ scenes. These scenes were captured using a Canon EOS 5D DSLR camera. Each scene consists of a high-quality combined image ($2,353$ images) with its corresponding DP views ($2,353 \times 2$ images). All images are captured at full-frame resolution (i.e., 6720×4480 pixels). Our data contains indoor and outdoor scenes with diverse image content, weather conditions, scene illuminations, and day/night scenes. We captured scenes with different aperture sizes (i.e., $f/4$, $f/5.6$, $f/10$, $f/16$, and $f/22$) in order to cover a wider range of spatially varying defocus blur (i.e., from all-in-focus to severely blurred images). We use this captured DP dataset in our multi-task framework to optimize directly for the DP-view synthesis task.

Other DP datasets We use the Canon DP deblurring dataset [3] (i.e., 350 training paired images) to optimize for both defocus deblurring and DP-view synthesis. We note that there is also a DP dataset based on the Google Pixel camera [25]. We opted to only use the Canon as it matches our dataset. Moreover, the Pixel DP data provides only the green channels in the raw-Bayer frame for the DP views.

7.3.4 Model Training

Our training is divided into two steps. First, training with image patches from our DP dataset is performed to optimize only the DP-view synthesis task. During this step the weights of the deblurring decoder branch ($\mathbf{Dec}_s$) are frozen. Once the model converges for the DP-view synthesis branches, the second step unfreezes the weights of $\mathbf{Dec}_s$ and starts fine-tuning using image patches from the Canon DP deblurring dataset [3] to optimize jointly for both the defocus deblurring and DP-view synthesis tasks (see training steps in Figure 4.2). For the first step, we train the network with the following loss terms:

$$\mathcal{L}_{ST1} = \mathcal{L}_{MSE}(l, r) + \mathcal{L}_C + \mathcal{L}_D, \tag{7.8}$$

where $\mathcal{L}_{ST1}$ is the overall first-step loss and $\mathcal{L}_{MSE}(l, r)$ is the typical MSE loss between the GT and estimated DP views. The second step needs more careful loss setting to fine-tune the model in a way that guarantees improving performance on both tasks. In the second step, the network is fine-tuned with the following loss terms:

$$\mathcal{L}_{ST2} = \mathcal{L}_{MSE}(s) + \lambda_1 \mathcal{L}_{MSE}(l, r) + \lambda_2 \mathcal{L}_C + \lambda_3 \mathcal{L}_D \tag{7.9}$$

where $\mathcal{L}_{ST2}$ is the overall second-step loss and $\mathcal{L}_{MSE}(s)$ is the typical MSE between the output deblurred image and the GT. The λ terms are added to control the training process.

While our novel losses (i.e., $\mathcal{L}_C$ and $\mathcal{L}_D$) can be reformulated and have a similar mathematical quadratic form of $\mathcal{L}_{MSE}(s)$ and $\mathcal{L}_{MSE}(l, r)$, the key point is that the new set of losses emphasizes important features (e.g., front/back focus ambiguity and flipping half-circle DP PSF shape). This new loss set is derived from the physical model of dual-pixel image formation and gives the proper guidance to capture those basic features. Therefore, the novel part is that how the loss terms are weighted, in which the weighting is naturally derived from real observation and PSF measures.

Table 7.1: Quantitative comparisons with single-image defocus deblurring methods. The best results are indicated with boldface. Results are on the Canon DP deblurring dataset [3] (test set consists of 37 indoor and 39 outdoor scenes).

Method	Indoor			Outdoor			Indoor & Outdoor				
	PSNR ↑	SSIM ↑	MAE ↓	PSNR ↑	SSIM ↑	MAE ↓	PSNR ↑	SSIM ↑	MAE ↓	NIQE ↓	Time ↓
JNB [70]	26.73	0.828	0.031	21.10	0.608	0.064	23.84	0.715	0.048	5.11	843.1
EBDB [67]	25.77	0.772	0.040	21.25	0.599	0.058	23.45	0.683	0.049	5.42	929.7
DMENet [68]	25.70	0.789	0.036	21.51	0.655	0.061	23.55	0.720	0.049	4.85	613.7
DPDNet (single) [3]	26.54	0.816	0.031	22.25	0.682	0.056	24.34	0.747	0.044	4.06	**0.5**
Our MDP	**28.02**	**0.841**	**0.027**	**22.82**	**0.690**	**0.052**	**25.35**	**0.763**	**0.040**	**3.25**	**0.5**

Table 7.2: Our single-image defocus deblurring achieves on-par results compared to DPDNet [3], which requires two DP images data as input. Results are on the Canon DP deblurring dataset [3].

Method	PSNR ↑	SSIM ↑	MAE ↓
DPDNet (real DP views) [3]	25.13	**0.786**	0.041
DPDNet (our synth. DP views) [3]	24.91	0.758	0.043
Our MDP (single image)	**25.35**	0.763	**0.040**

7.4 Experiments

7.4.1 Training Details

Training data We divide our newly captured DP dataset into 2,090 and 263 training and testing scenes, respectively. In the first training step, we use the 2,090 training scenes. For the second training step, we use the DP data from [3] following the same data division in [3]—that is, 350, 74, and 76 training, validation, and testing scenes, respectively.

Training procedure We extract image patches of size $512 \times 512 \times 3$, where the input is a single patch and the output is three patches. The convolutional layer weights are initialized using He's method [100] and the Adam optimizer [90] is used to train the model. The mini-batch size in each iteration is set to 8 batches.

For the first training step, the initial learning rate is set to 3×10^{-4}, which is decreased by half every 8 epochs. The model converges after 60 epochs in the first step. For the second step, the initial learning rate is set to 6×10^{-4}, which is decreased by half every 8 epochs. The model converges after 80 epochs. The λ terms are set to 0.8, 0.5, and

0.5, respectively, in order to have a balanced loss minimization and to guide the network attention towards minimizing for defocus deblurring in the second step. All the training details and model setup are implemented using Python with the Keras [101] framework on top of TensorFlow [102] and trained with NVIDIA TITAN X GPU.

7.4.2 Single-Image Deblurring

To evaluate our method, we use the test set of the Canon DP deblurring dataset [3]. Specifically, this test set [3] includes 37 indoor and 39 outdoor scenes. We compare our results against recent methods for single-image defocus deblurring [67,68,70]. For the sake of completeness, we also compare our results against the recent DP defocus deblurring method (DPDNet) [3], which requires the availability of DP data at inference time. As our task is single-image defocus deblurring, we provide the results of the DPDNet [3] using single input images for a fair comparison. The single-image model of this method was originally trained in [3] (Chapter 5) with the same model size (i.e., number of weights) used for the DP model. Although our MDP model has three decoder branches, we adjust the number of convolutional operations and filter size in some blocks in order to have an equivalent model size in terms of number of weights compared to DPDNet [3]. Our proposed MDP is fully convolutional so that we can test on full-size images regardless of the patch size used for training.

Quantitative results Table 7.1 shows the quantitative results of our method and other single-image defocus deblurring methods: the just noticeable defocus blur method (JNB) [70], the edge-based defocus blur estimation method (EBDB) [67], the deep defocus map estimation method (DMENet) [68], and the DPDNet (single) [3]. We use the common signal processing metrics PSNR, SSIM, and MAE. We also report the Naturalness Image Quality (NIQE) metric of the output deblurred images with respect to a reference model derived from the DP GT images. As shown in Table 7.1, our method achieves the state-of-the-art results for all metrics compared to other recent single-image defocus deblurring methods. Furthermore, MDP and DPDNet (single) have much lower inference time—that is, >1,200× faster compared to other methods.

Though motion blur leads to image blur too, as defocus blur does, the physical formation and consequently the appearance of the resultant blur are different. This was noted in [3], and we found a notable degradation on the accuracy of methods focused on

motion blur [2, 15, 17, 18, 21] when they are applied to defocus blur; for example, Tao et al.'s method [2] for motion deblurring achieves an average PSNR of 20.12dB when it is evaluated on the Canon DP deblurring test set [3], which is notably lower than all other defocus deblurring methods shown in Table 7.1.

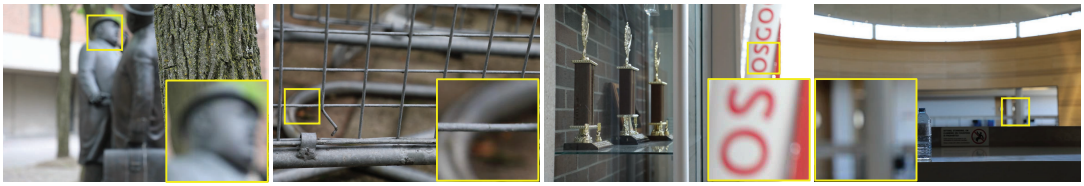
In Table 7.1, we reported the results of the DP-based method (i.e., DPDNet) [3] trained on single input. For the sake of completeness, we also compared our method against this method when it is fed with real DP data as input. Table 7.2 shows this comparison. As can be seen, our method achieves higher PSNR and MAE but lower SSIM compared to DPDNet [3], while our method is more practical as it requires only single-input images compared to the DPDNet [3], which requires accessing the two DP images at the inference phase. Recall that DPDNet cannot be trained on our DLDP dataset as it is unpaired and does not have the corresponding GT all-in-focus image. This point is a central strength of our approach, as the use of unpaired DP images enables capturing a much larger dataset for training, thereby making the learning process more efficient.

Qualitative results Our method achieves better qualitative results when compared with several existing single-image defocus deblurring methods (as shown in Figure 7.1 and Figure 7.6). We provide additional qualitative comparisons in Figure 7.5, where we compare our results against the results of the EBDB [67], DMENet [68], and the DPDNet (single) [3] methods. Figure 7.1 and Figure 7.6 show that our method achieves results that are arguably visually superior to the other methods.

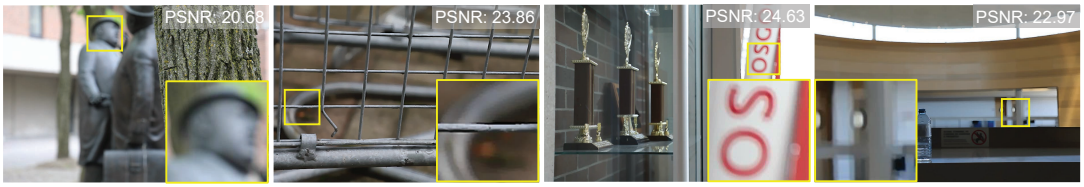
To demonstrate qualitatively the deblurring generalization ability of our proposed MDP, we tested MDP on other cameras as shown in Figure 7.7. In this experiment, MDP is trained only on the Canon data, but tested on images from other cameras.

7.4.3 Ablation Study

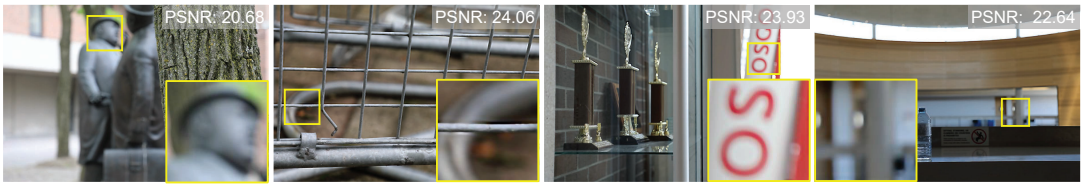
Multi-task effectiveness As explained in Section 7.3, our method is a multi-task framework that is suitable not only for reducing defocus blur but also for predicting DP views of the input single image. Our multi-task framework allows our method to improve the results of each task, as they are inherently correlated. Intuitively, we can re-design our framework by training a single model for each task separately. Table 7.3 shows the results of training a single model (with approximately the same capacity of our multi-task framework) on each task separately. Table 7.3 also shows the results of training both single



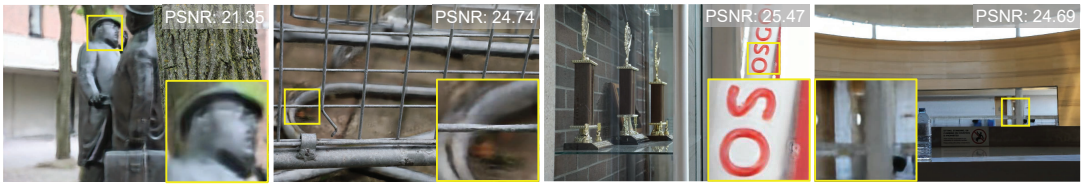
(a) Input



(b) EBDB [67]



(c) DMENet [68]



(d) DPDNet (single) [3]

(e) Ours

(f) Ground truth

Figure 7.5: Qualitative comparison with single-image defocus deblurring methods on the test set of the Canon DP deblurring dataset [3]. Note that DPDNet was originally introduced to use DP images as input, but the authors in [3] also provided the same model trained on a single image, denoted as DPDNet (single). Our method produces the best quantitative and arguably qualitative results.

(a) Input

(b) EBDB [67]

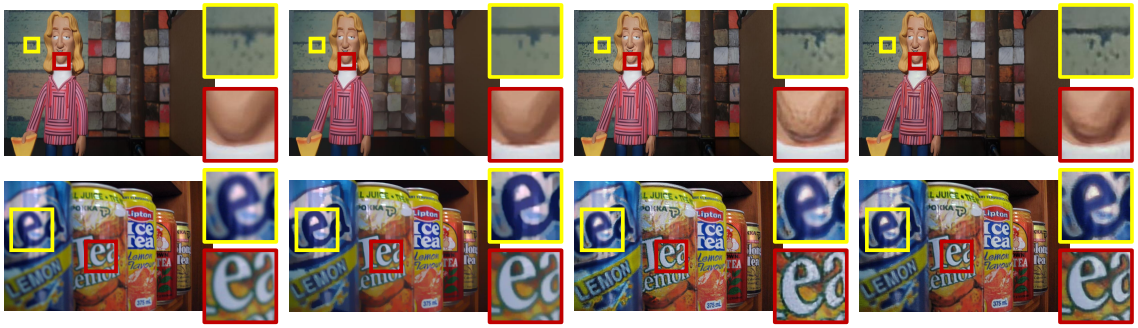
(c) DMENet [68]

(d) DPDNet (single) [3]

(e) Ours

(f) Ground truth

Figure 7.6: Additional qualitative comparisons with other single-image defocus deblurring methods on the test set of the Canon DP dataset [3]. Note that DPDNet was originally introduced to use DP images as input, but the authors in [3] also provided the same model trained on a single image, denoted as DPDNet (single). Our method produces the best quantitative and arguably best qualitative results.



(a) Input (b) EBDB [67] (c) DPDNet (single) [3] (d) Ours

Figure 7.7: Qualitative comparison with single-image defocus deblurring methods on new camera devices (top: Samsung S5, bottom: Flickr image [erasmusa CC BY-NC 2]. We compare our results with the following single-image defocus deblurring methods: EBDB [67] and DPDNet (single) [3]. Note that DPDNet was originally introduced to use DP images as input, but the authors in [3] also provided the same model trained on a single image, denoted as DPDNet (single). Our method generalizes well for unseen cameras during the training stage and produces arguably best qualitative results compared with other methods.

Table 7.3: Ablation study to demonstrate the effectiveness of our DP-based loss terms and multi-task training. Results are on the Canon DP deblurring dataset [3]. Note that the $\mathcal{L}_C$ and $\mathcal{L}_D$ are not applicable (N/A) to the single task defocus deblurring as it does not predict the DP views.

Method	Defocus deblurring		DP-pixel synthesis	
	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
Single-task w/o $\mathcal{L}_C$ and $\mathcal{L}_D$	24.34	0.747	37.05	0.953
Single-task w/ $\mathcal{L}_C$ and $\mathcal{L}_D$	N/A	N/A	38.23	0.962
Multi-task w/o $\mathcal{L}_C$ and $\mathcal{L}_D$	24.81	0.750	38.01	0.957
Multi-task w/ $\mathcal{L}_C$ and $\mathcal{L}_D$	**25.35**	**0.763**	**39.17**	**0.973**

and multi-task frameworks with and without our DP-based loss functions introduced in Section 7.3.2. As shown, our multi-task framework with our introduced loss functions achieves the best results.

Multi-decoder Stitching We also investigate the utility of having multiple weight sharing stages by introducing a variation of MDP network with different multi-decoder stitching options: (1) *no stitching* that makes the latent space $\mathcal{X}$ the only weight sharing stage, (2) *late-stage stitching* at the last block, and (3) the original proposed MDP with *middle-stage stitching*. We report the results of MDP variations in Table 7.4. The training procedure followed is the same as for all MDP variations as described in Section 7.4.1.

The results in Table 7.4 show that *middle-stage stitching* achieves the best results as it allows weight sharing at multiple stages compared with the *no stitching* variation. On the other hand, there is a noticeable drop in the deblurring performance when *late-stage stitching* is applied as the sharpness of the deblurring decoder (i.e., $\mathbf{Dec}_s$) is affected by the half-PSF blur present in feature maps of the synthesized DP views (i.e., $\mathbf{Dec}_l$ and $\mathbf{Dec}_r$) at this later stage.

7.4.4 DP-View Synthesis and Application

Reflection removal As discussed in Section 7.1, DP data has been used for different computer vision tasks. Here we show using our synthesized DP views can be leveraged for a DP-based method in the absence of real DP data. We validate this idea of using our reconstructed DP views as a proxy to DP data on the reflection removal and defocus

Table 7.4: This table reports results on an ablation study performed to examine the effectiveness of the multi-decoder stitching design for defocus deblurring. Three variations of our proposed MDP based on three different stitching options: (1) no stitching, (2) late-stage stitching and (3) middle-stage stitching. Results reported are on the Canon DP deblurring dataset [3].

MDP variation	PSNR ↑	SSIM ↑	MAE ↓
MDP (no stitching)	25.03	0.757	0.042
MDP (late-stage stitching)	25.16	0.759	0.041
MDP (middle-stage stitching)	**25.35**	**0.763**	**0.040**

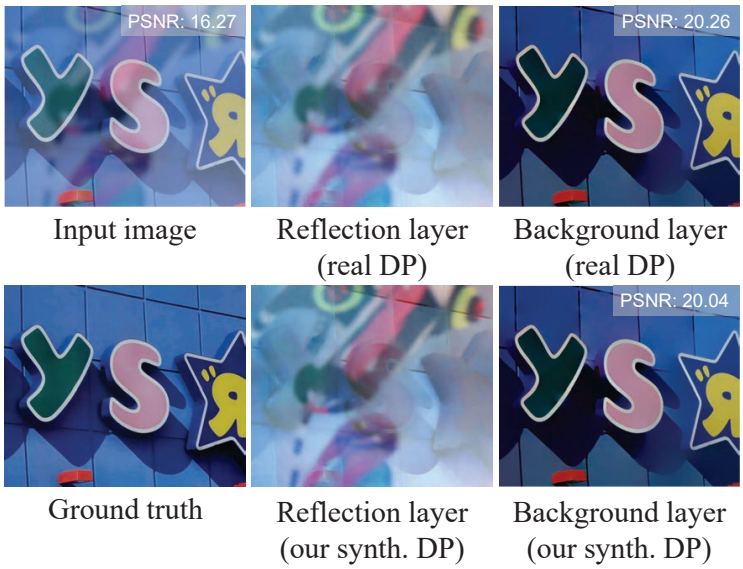


Figure 7.8: Our synthetic DP views can be used as input for DP-based reflection removal of [5], and are able to produce results on par with those using real DP data.

deblurring tasks. Specifically, we processed input real DP data and our generated DP data using the DP-based reflection removal (DPRR) [5] and defocus deblurring (DPDNet) [3] methods. As shown in Figure 7.8, utilizing our synthetic DP views produces approximately the same high-quality result as using DPRR [5] on real DP data. This allows us to achieve better reflection removal results, while still requiring only *a single input image*, compared to other methods for reflection removal, as shown in Table 7.5. As for DPDNet, Table 7.2 shows that DPDNet tested with our generated DP views has on-par results compared to the one tested with real DP views.

Table 7.5: Reflection removal quantitative results on the dataset proposed in [5]. When using our synthetic DP views, the dual-pixel reflection removal (DPRR) method [5] achieves on-par results compared with using real DP views, which makes the DPRR method applicable with the absence of real DP data.

Single-image		Non-single-image	
Method	PSNR	Method	PSNR
ZN18 [131]	15.57	LB13 [132]	16.12
YG18 [133]	16.49	GC14 [134]	16.02
DPRR [5] (ours)	**19.32**	DPRR [5] (real DP)	**19.45**

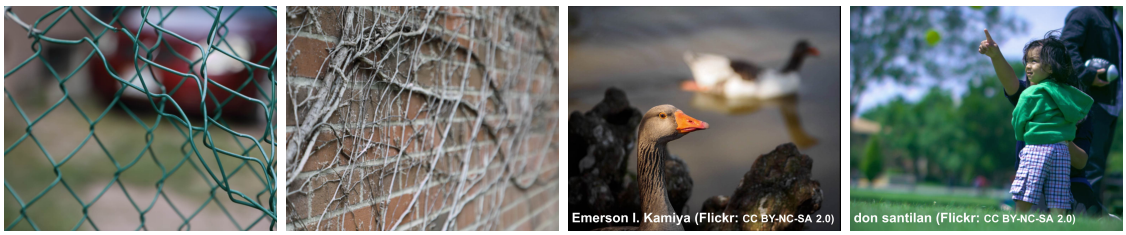
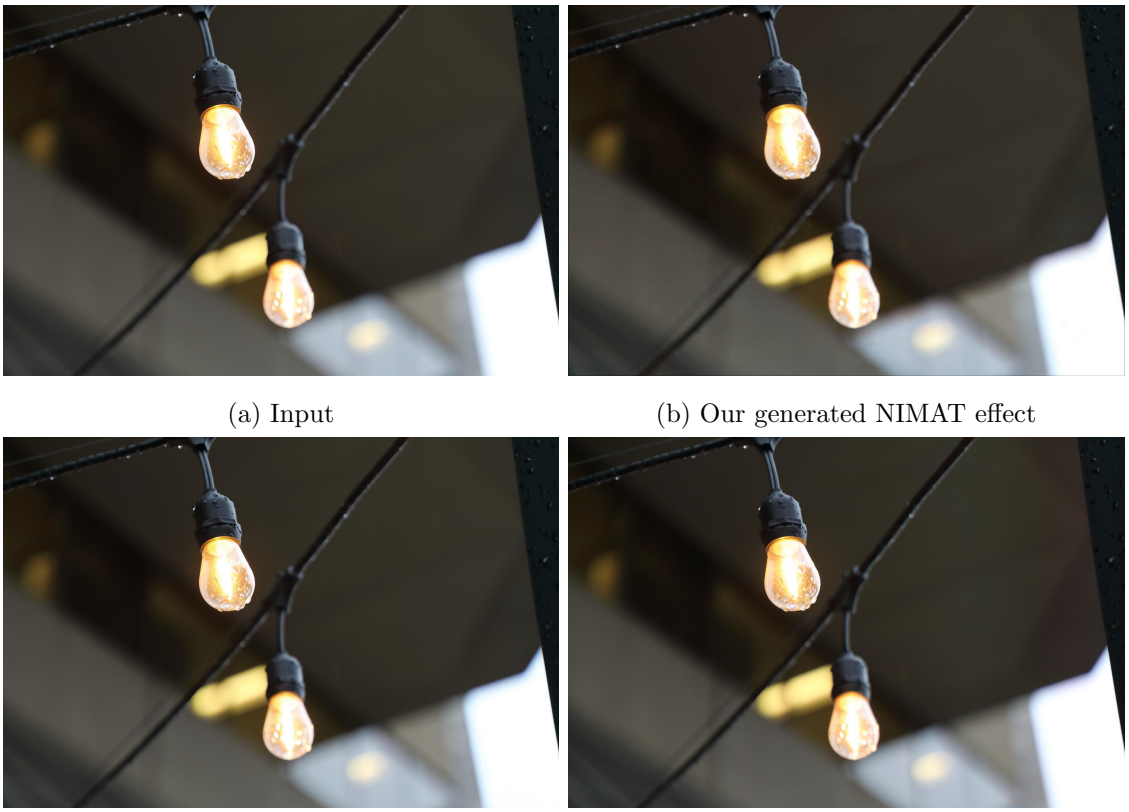


Figure 7.9: Our novel NIMAT effect. Aesthetically pleasing image motion produced by our multi-view synthesis. The first two input images were taken from the Canon DP deblurring dataset [3], while the others are from Flickr.**Note: these are animated figures; click on the figure to start the animation. It is recommended to open this PDF in Adobe Acrobat Reader to work properly.**

Image motion effect An interesting side effect of our multi-task network is the ability to perform view synthesis. For instance, we can generate an aesthetically realistic image motion by synthesizing a multi-view version of a given single image. As discussed in Section 7.2, the DP two sub-aperture views of the scene depend on the sensor's orientation and, in this chapter, our dataset contains left/right DP pairs, and consequently our network synthesizes the horizontal DP disparity. We can synthesize additional views with different "DP disparity" by rotating the input image before feeding it to our network by a 45° clockwise step three times (i.e., 45°, 90°, 135°). This allows us to produce a smooth motion from the reconstructed eight views as shown in Figure 7.9. We refer to this DP-based view synthesis and image motion as *new image motion attribute* (NIMAT) effect.

In Figure 7.10, Figure 7.11, Figure 7.12, and Figure 7.13, we also provide examples

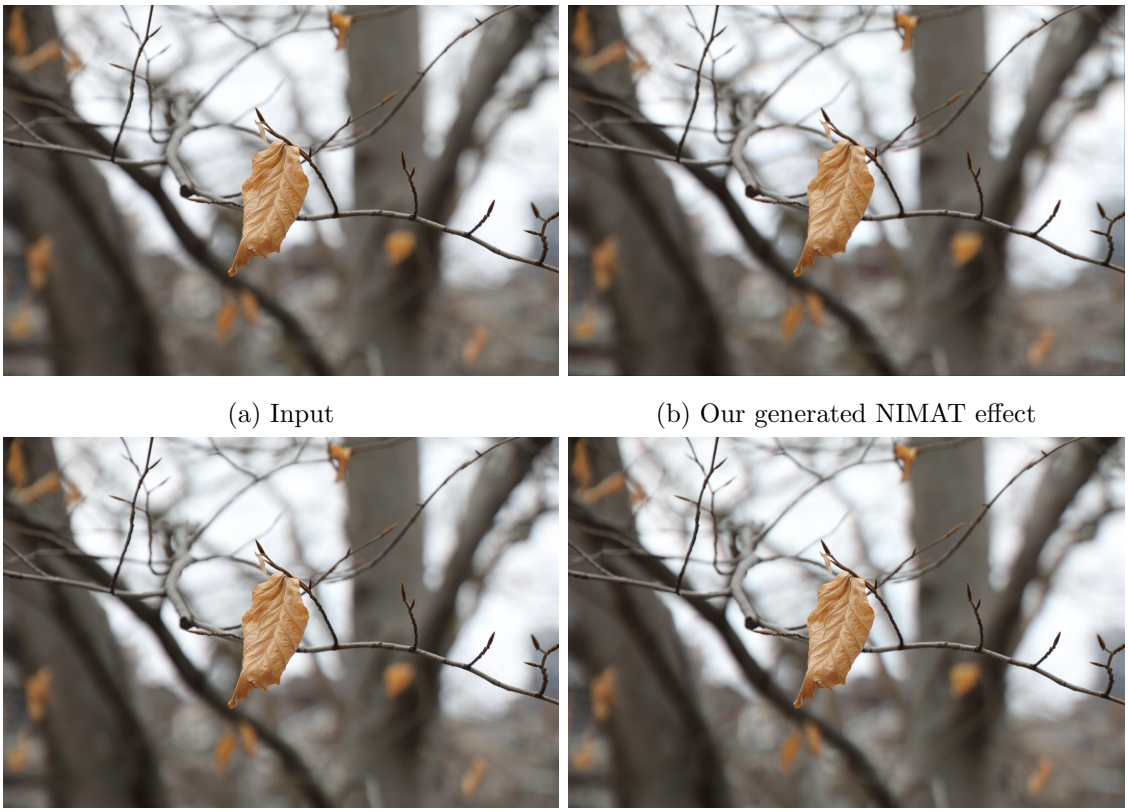


(a) Input (b) Our generated NIMAT effect

(c) Our DP views (d) GT DP views

Figure 7.10: An example from our newly captured DP dataset. (a) Input combined image $\mathbf{I}_c$. (b) Our novel NIMAT effect generated using the proposed MDP. (c) Animated results of our synthesized DP views. (d) Animated ground truth DP views. Our MDP is able to generate high-quality eight/DP views. **Note: (b), (c), and (d) are animated figures; click on the figure to start the animation. It is recommended to open this PDF in Adobe Acrobat Reader to work properly.**

from our newly captured DP dataset along with results of high-quality reconstructed DP views and image motion of the synthesized eight views (i.e., NIMAT effect). Recall that our MDP is trained using data captured by Canon 5D DSLR camera and Figure 7.14 shows qualitatively how our proposed MDP is able to generate high-quality eight views for other cameras. Based on the visual image motion results in Figure 7.14, our MDP is able to generalize well to other non-Canon cameras by generating a visually pleasing NIMAT effect.



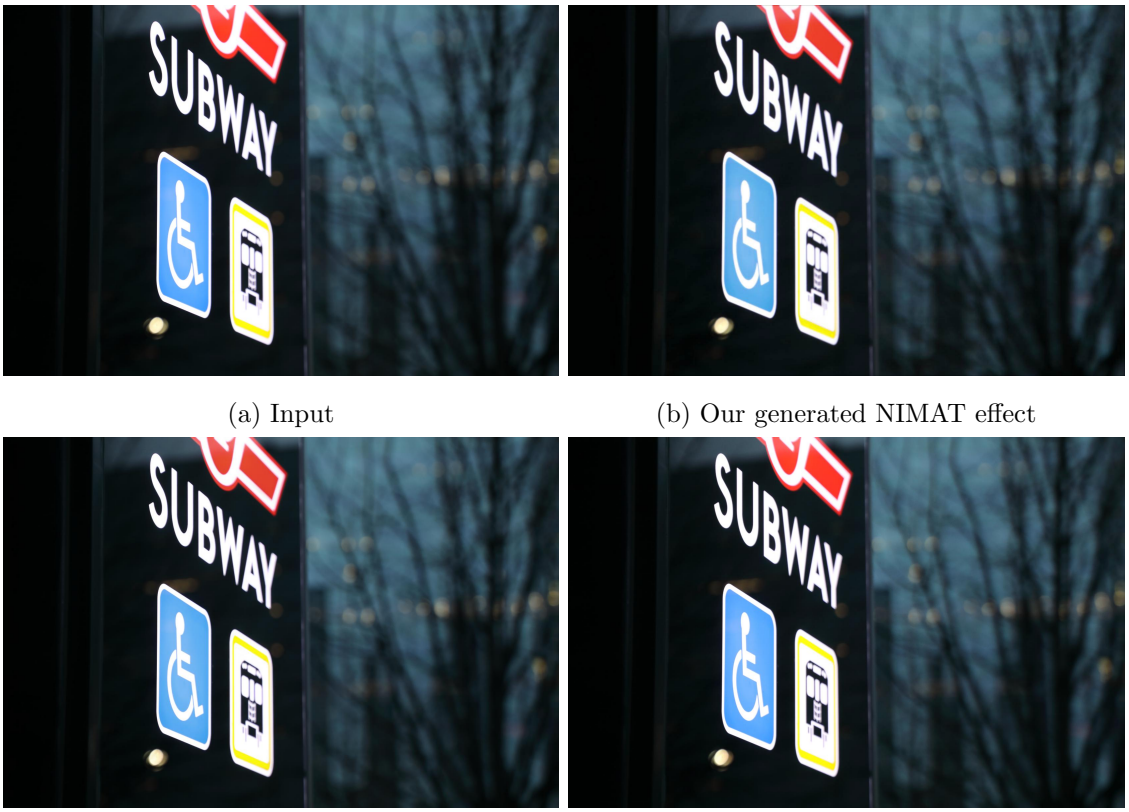
(a) Input (b) Our generated NIMAT effect

(c) Our DP views (d) GT DP views

Figure 7.11: Additional example from our newly captured DP dataset. (a) Input combined image $\mathbf{I}_c$. (b) Our novel NIMAT effect generated using the proposed MDP. (c) Animated results of our synthesized DP views. (d) Animated ground truth DP views. Our MDP is able to generate high-quality eight/DP views. **Note: (b), (c), and (d) are animated figures; click on the figure to start the animation. It is recommended to open this PDF in Adobe Acrobat Reader to work properly.**

7.5 Conclusion

We have demonstrated that a DNN trained for the purpose of single-image defocus deblurring can be improved by incorporating the additional task of synthesizing the two DP views associated with the input image. One benefit of this approach is that capturing data for the DP view synthesis task is easy to perform and requires no special capture setup. This is contrasted with a conventional approach that requires careful capture of sharp/blurred image pairs for the deblurring task. This multi-task strategy is able to

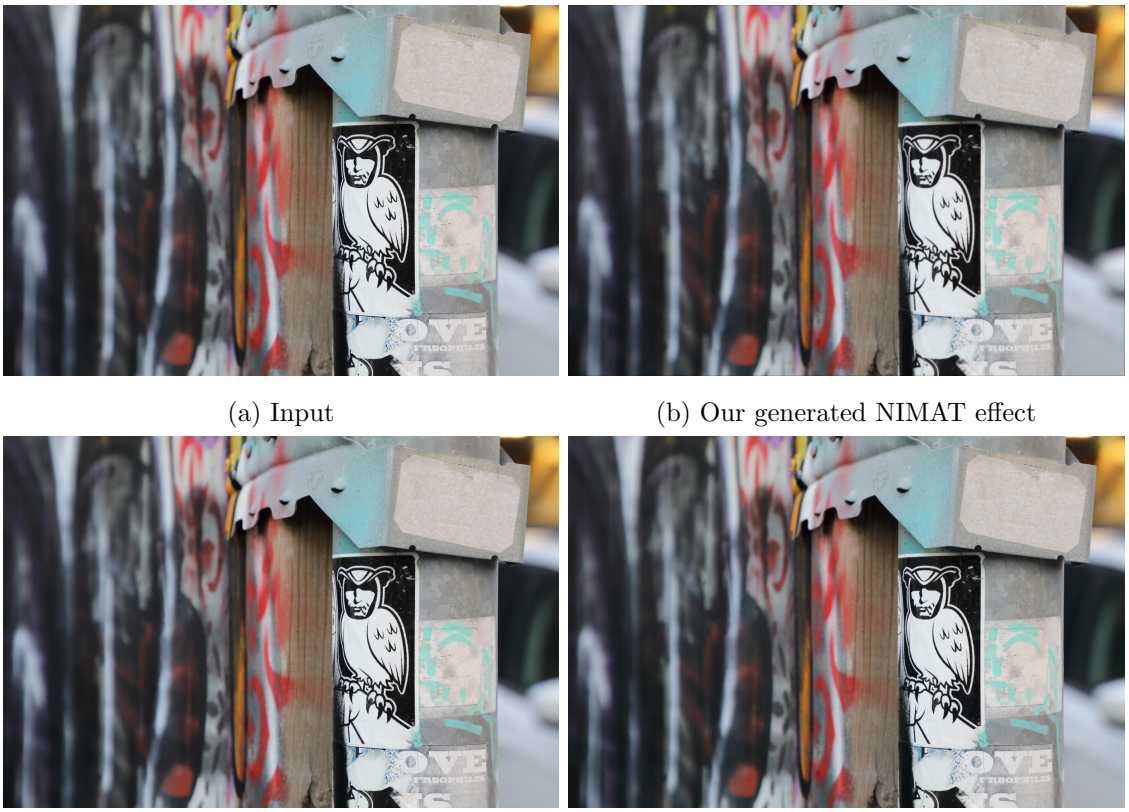


(a) Input (b) Our generated NIMAT effect

(c) Our DP views (d) GT DP views

Figure 7.12: Additional example from our newly captured DP dataset. (a) Input combined image $\mathbf{I}_c$. (b) Our novel NIMAT effect generated using the proposed MDP. (c) Animated results of our synthesized DP views. (d) Animated ground truth DP views. Our MDP is able to generate high-quality eight/DP views. **Note: (b), (c), and (d) are animated figures; click on the figure to start the animation. It is recommended to open this PDF in Adobe Acrobat Reader to work properly.**

improve deblurring results by close to 1dB in terms of PSNR. As an added benefit, we introduced the novel NIMAT effect and showed that our DNN is able to perform realistic view synthesis that can be used for tasks such as reflection removal. Our dataset, code, and trained models are publicly available at: `https://github.com/Abdullah-Abuolaim/multi-task-defocus-deblurring-dual-pixel-nimat`.



(a) Input (b) Our generated NIMAT effect

(c) Our DP views (d) GT DP views

Figure 7.13: Additional example from our newly captured DP dataset. (a) Input combined image $\mathbf{I}_c$. (b) Our novel NIMAT effect generated using the proposed MDP. (c) Animated results of our synthesized DP views. (d) Animated ground truth DP views. Our MDP is able to generate high-quality eight/DP views. **Note: (b), (c), and (d) are animated figures; click on the figure to start the animation. It is recommended to open this PDF in Adobe Acrobat Reader to work properly.**

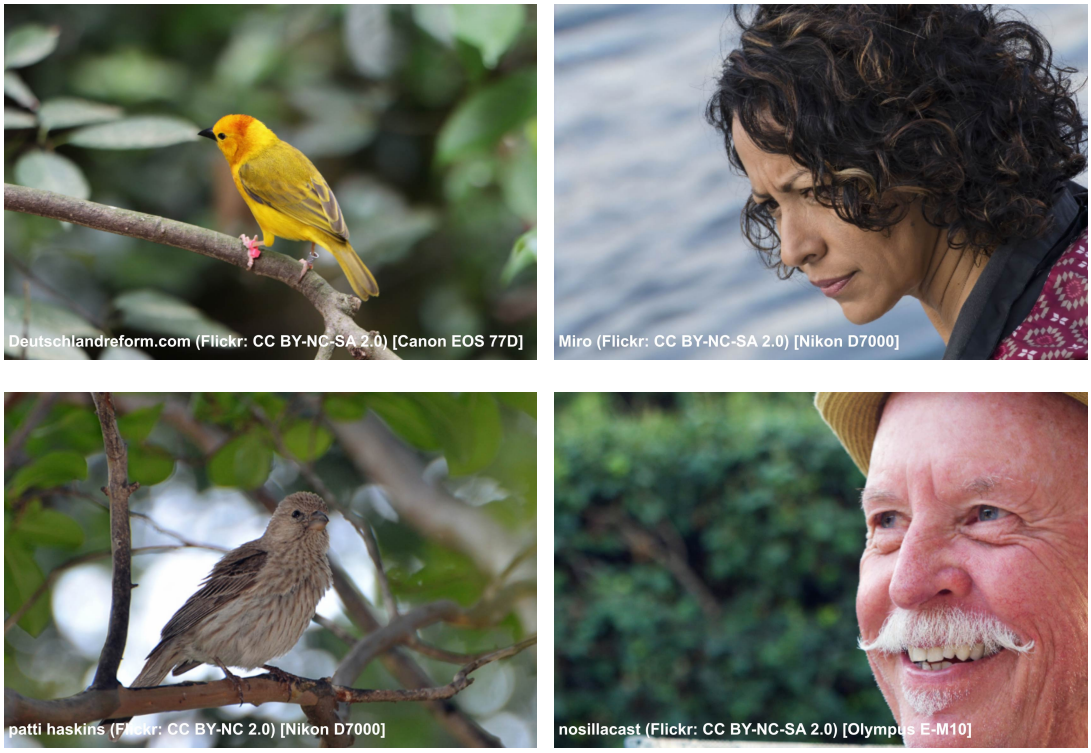


Figure 7.14: Our novel NIMAT effect. Multi-view synthesis results of our proposed MDP applied to other cameras than the Canon 5D DSLR camera (used for training). These results are synthesized from a single input image captured by new camera devices, in which they do not have the ground truth DP views. Our MDP produces high-quality eight views that can be used to create an aesthetically pleasing NIMAT effect. Furthermore, these results demonstrate a good generalization ability of our MDP as it can provide high-quality views from images that are captured by unseen camera device during the training stage. **Note: these images are animated; click on the image to start the animation. It is recommended to open this PDF in Adobe Acrobat Reader to work properly.**

Chapter 8

Synthetic Shallow DoF

In the previous chapters (i.e, Chapter 5, Chapter 6, Chapter 7), we addressed the problem of extending camera DoF as a post-processing step. This chapter examines another way of manipulating camera DoF (e.g., shallow DoF) by applying synthetic defocus blur based on a real camera model. This type of DoF manipulation is known as *synthetic bokeh* effect and widely employed in the recent smartphone portrait mode.

Portrait mode is widely available on smartphone cameras to provide an enhanced photographic experience. One of the primary effects applied to images captured in portrait mode is a synthetic shallow depth of field (DoF). The synthetic DoF (or bokeh effect) selectively blurs regions in the image to emulate the effect of using a large lens with a wide aperture. In addition, many applications now incorporate a new image motion attribute (NIMAT) to emulate background motion, where the motion is correlated with estimated depth at each pixel. In this chapter, we follow the trend of rendering the NIMAT effect by introducing a modification on the blur synthesis procedure in portrait mode. In particular, our modification enables a high-quality synthesis of multi-view bokeh from a single image by applying rotated blurring kernels. Given the synthesized multiple views, we can generate aesthetically realistic image motion similar to the NIMAT effect. We validate our approach qualitatively compared to the original NIMAT effect and other similar image motions, like Facebook 3D image. Our image motion demonstrates a smooth image view transition with fewer artifacts around the object boundary.

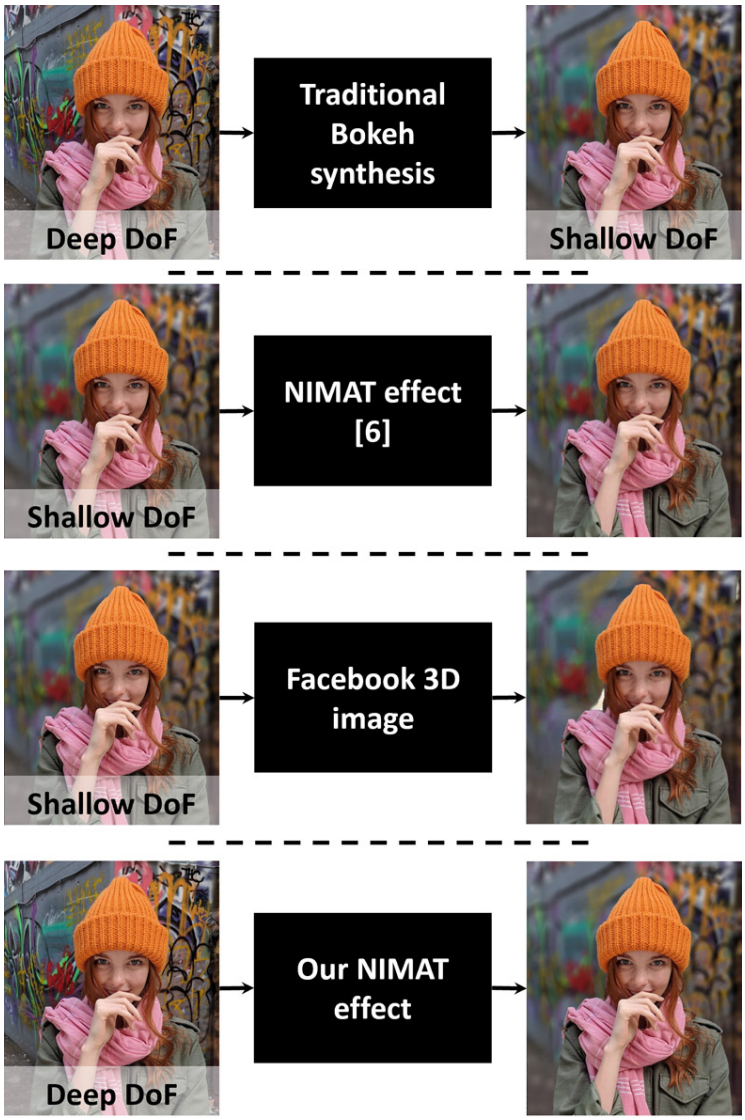


Figure 8.1: This figure shows a comparison between different image motion effects. We also show the output of the traditional bokeh synthesis. Our approach takes the sharp image (i.e., deep DoF) to generate the image motion. Other approaches start with the blurry input (i.e., shallow DoF) to synthesize the image motion. **Note: this figure is animated; click on the image to start the animation. It is recommended to open this PDF in Adobe Acrobat Reader to work properly.**

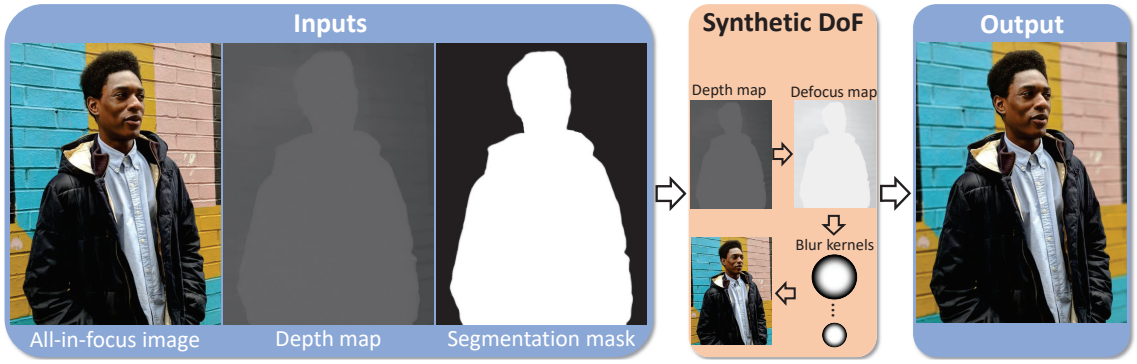


Figure 8.2: This figure shows a typical synthetic shallow depth of field (DoF) processing framework. This framework takes three inputs: single image, estimated depth map, and segmentation mask. Given the inputs, the synthetic DoF unit produces the desired image. The image, depth map, and segmentation mask are taken from the dataset in [27].

8.1 Introduction

Unlike digital single-lens reflex (DSLR) and mirrorless cameras, smartphone cameras cannot produce a natural shallow depth of field (DoF) due to the camera's small aperture and simple optical system. Instead, many smartphones (e.g., iPhone 12, Google Pixel 4, Samsung Galaxy) emulate a shallow DoF via a *portrait mode* setting that processes the image at capture time. These methods typically isolate the subject from the background and then blur the background to emulate the swallow DoF [27]. An example is shown in the first row of Figure 8.1.

Most smartphone cameras apply the synthetic bokeh effect using a common image processing framework. This traditional procedure takes an input image with minimal DoF blur and an estimated depth map to determine the blur kernel size at each pixel (i.e., defocus map). In some cases, a segmentation mask is also used to avoid blurring pixels that belong to the people and their accessories. Figure 8.2 shows an illustrative example of the common synthetic bokeh framework.

Recently, Abuolaim et al. proposed a new image motion attribute (NIMAT) effect [127] that generates multiple sub-aperture views based on DoF blur and dual-pixel (DP) image formation. Abuolaim et al.'s method produces multiple views from a single input image captured by a DSLR camera and has a natural shallow DoF. Their DP- and DoF-based view synthesis is designed to generate pixel motion correlated to the defocus blur size at

each pixel. However, obtaining an image with a natural shallow DoF using a smartphone camera is difficult, as mentioned earlier. Inspired by NIMAT [127], we provide a similar effect by modifying the traditional synthetic bokeh framework. Our modification enables synthesizing shallow DoF along with generating multiple views by applying a rotated blurring kernel. In our proposed framework, the defocus blur kernel shape is determined based on the sub-aperture image formation found in DP sensors. To our knowledge, we are the first to introduce this novel synthetic bokeh and DP-/DoF-based multi-view synthesis. Figure 8.1 shows a comparison of different image motion approaches. It also provides the output of the traditional bokeh synthesis in the first row. Recall that other image motion approaches do not synthesize the bokeh effect. As a result, our method combines image motion and synthetic DoF into a single step. As demonstrated in Figure 8.1, our image motion exhibits a smooth view transition with fewer artifacts around the object boundary compared to other approaches. The contributions made in this chapter were published in the Workshop on Applications of Computational Imaging (WACI) that is organized in conjunction with the Winter Conference on Applications of Computer Vision (WACV), 2022 [135].

8.2 Releated Work

Synthetic bokeh The bokeh effect in photography is an aesthetic quality of the blur that renders the main subject of the taken photo in focus while the background details fall out of focus. As mentioned earlier, standard smartphone cameras cannot produce such bokeh photographs due to the small size of the aperture and short focal length used in almost all smartphone cameras. Due to this limitation, a large body of work has targeted ways to emulate a shallow DoF image for smartphone cameras (e.g., [27, 80, 82, 136–139]).

Prior methods require either up-down translation of the camera (e.g., [136]) or benefits from the parallax caused by accidental handshake during capturing (e.g., [137, 138]). However, both strategies may lead to undesirable results as they rely on a specific type of movement that is not always applied in real scenarios. As a result, having low parallax limits these methods' ability to work properly.

Another strategy requires multi-image capturing, or stereo imaging, to estimate image depth from defocus cues extracted from these multiple images, or stereo pairs, of the same scene [80, 82, 140–143]. However, this strategy results in ghosting effects and cannot work

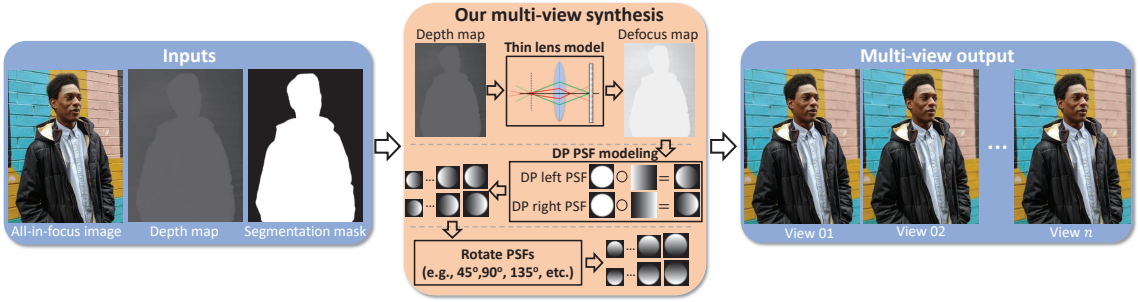


Figure 8.3: An overview of our proposed framework for multi-view synthesis based on rotated DP blur kernels. This framework takes three inputs: single image, estimated depth map, and segmentation mask. Given the inputs, the multi-view synthesis unit produces n views based on the number of rotated PSFs. The image, depth map, and segmentation mask are taken from the dataset in [27].

properly with non-static objects.

Instead of relying on multi-image capture, monocular single-image depth estimation methods are adopted to predict depth information using either inverse rendering [144,145] or supervised machine learning [146–149]. Given the estimated depth map, synthetic rendering of shallow DoF images is then a straightforward process. However, the quality of this synthetic bokeh effect is tied to the accuracy of the estimated depth map. In recent years, learning-based depth estimation methods have achieved impressive results; however, like most deep learning-based techniques, such learning depth estimators often suffer from poor generalization to images taken under conditions beyond training examples. Thus, synthesized shallow DoF images could suffer from obvious artifacts around the main object's edges.

To mitigate failure cases in single-image depth estimation, a few methods propose to replace the depth estimation process with some constraints in the scene to improve the results. For example, by dealing only with photos of people against a distant background, bokeh effects can be generated without a need for a depth map estimation [150,151]. With this reasonable constraint, synthetic shallow DoF can be achieved by first segmenting out the human subject. This is typically performed using a trained convolutional neural network. Next, the background can be blurred using a global blur kernel. While effective, this approach assumes a constant difference in depth between the main subject (i.e., people) and the background. In addition, this approach requires a deep network to segment people

from images properly.

Unlike all methods above, in this chapter, our goal is to produce an image motion effect similar to the NIMAT effect [127]. A high-quality bokeh synthesis is an extra by-product output.

Dual-pixel sensor DP sensors were developed as a means to improve the camera's autofocus system. The DP design produces two sub-aperture views of the scene that exhibit differences in phase that are correlated to the amount of defocus blur. Then, the phase difference between the left and right sub-aperture views of the primary lens is calculated to measure the blur amount. The phase information is also used to adjust the camera's lens such that the blur is minimized. While intended for autofocus [14, 23], the DP images have been found useful for other tasks, such as depth map estimation [4,25,126], defocus deblurring [3, 109, 110], and synthetic DoF [27].

8.3 Defocus-Based Multi-View Synthesis

In this section, we describe our framework for multi-view synthesis based on rotated DP blur kernels. An overview of the proposed framework is shown in Figure 8.3. First, we introduce the thin lens model used to determine the blur kernel size at each pixel. Then, the DP point spread function (PSF) is described in Section 8.3.2. Afterward, Section 8.3.3 introduces the defocus blur procedure. Lastly, Section 8.3.4 explains the process of multi-view synthesis via rotated PSFs.

8.3.1 Blur Kernel Size Based on the Thin Lens Model

The size of the PSFs at each pixel in the image can be calculated using the depth map. Therefore, we model camera optics using a thin lens model that assumes negligible lens thickness, helping to simplify optical ray tracing calculations [112]. This model can approximate the circle of confusion (CoC) size for a given point based on its distance from the lens and camera parameters (i.e., focal length, aperture size, and focus distance). This model is illustrated in Figure 8.4, where f is the focal length, s is the focus distance, and d is the distance between the scene point and camera lens. The distance between the lens

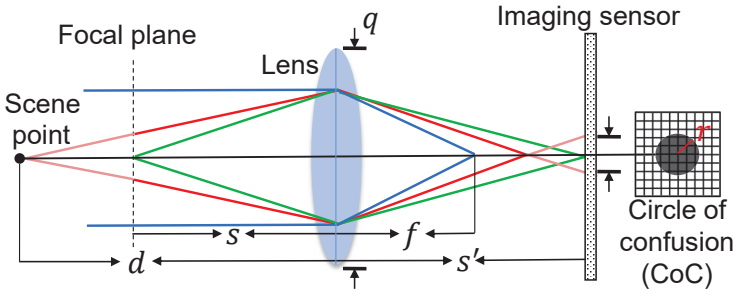


Figure 8.4: Thin lens model illustration and dual-pixel image formation. The circle of confusion (CoC) size is calculated for a given scene point using its distance from the lens, camera focal length, and aperture size. Note: we acknowledge that this figure was adapted from [110]

.

and sensor s', and the aperture diameter q are defined as:

$$s' = \frac{f\ s}{s - f}, \tag{8.1}$$

$$q = \frac{f}{F}, \tag{8.2}$$

where F is the f-number ratio. Then, the CoC radius r of a scene point located at distance d from the camera is:

$$r = \frac{q}{2} \times \frac{s'}{s} \times \frac{d - s}{d}. \tag{8.3}$$

8.3.2 Blur Kernel Shape Based on Dual-pixel Image Formation

Once the radius of the PSF is calculated at each pixel (Section 8.3.1), we need to decide the PSF shape to be applied. In this section, we adopt a DP-based PSF shape for DP view synthesis.

We start with a brief overview of DP sensors. A DP sensor uses two photodiodes at each pixel location with a microlens placed on the top of each pixel site, as shown in Figure 8.5-c. This design was developed by Canon to improve camera autofocus by functioning as a simple two-sample light field camera. The two-sample light-field provides two sub-aperture views of the scene and, depending on the sensor's orientation, the views can be referred to as left/right or top/down pairs; we follow the convention of prior work [3, 4] and refer to them as the left/right pair. The light rays coming from scene points that are within

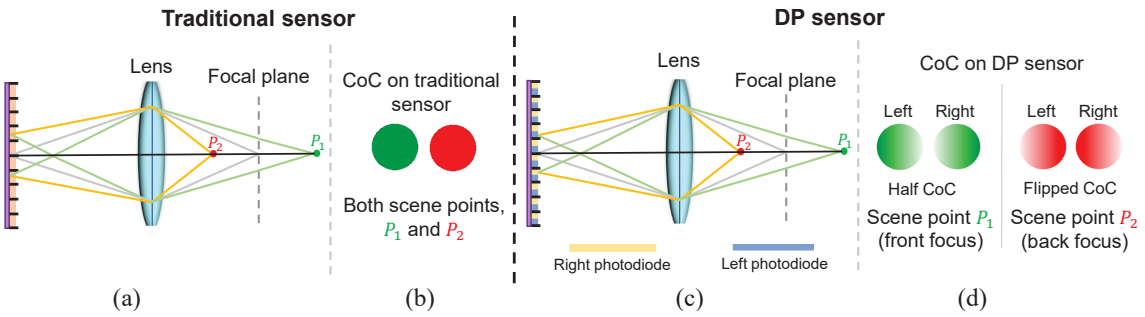


Figure 8.5: Circle of confusion (CoC) formation in DP sensors. (a) Traditional sensor and (c) DP sensor. (b) and (d) are the CoC formation on the 2D imaging sensor of two scene points, $P1$ and $P2$. On the two DP views, the half-CoC flips direction if the scene point is in front or back of the focal plane. Note: we acknowledge that this figure was adapted from [127].

the camera's DoF exhibit little to no difference in phase between the views. On the other hand, light rays coming from scene points outside the camera's DoF exhibit a noticeable defocus disparity in the *left-right* views. The amount of defocus disparity is correlated to the amount of defocus blur.

Unlike traditional stereo, the difference between the DP views can be modeled as the latent sharp image being blurred in two different directions using a half-circle PSF [4]. This is illustrated in the resultant CoC of Figure 8.5-d. The ideal case of a half-circle CoC on real DP sensors is only an approximation due to constraints of the sensor's construction and lens array. These constraints allow a part of the light ray bundle to leak into the other-half dual pixels (see half CoC of left/right views in Figure 8.5-d).

Unlike other approaches [4,110], we provide a simplified model of the DP PSF using a disk $\mathbf{C}$ shape that is element-wise multiplied by a ramp mask as follows:

$$\mathbf{H}_l = \mathbf{C} \circ \mathbf{M}_l, \quad \text{s.t.} \quad \mathbf{H}_l \geq \mathbf{0}, \text{ with} \sum \mathbf{H}_l = 1, \tag{8.4}$$

where $\circ$ denotes element-wise multiplication, $\mathbf{M}_l$ is a 2D ramp mask with a constant intensity fall-off towards the right direction, and $\mathbf{H}_l$ is the left DP PSF. One interesting property of the DP sensors is that the right DP PSF $\mathbf{H}_r$ is the $\mathbf{H}_l$ that is flipped around the vertical axis – namely, $\mathbf{H}_l^f$:

$$\mathbf{H}_r = \mathbf{H}_l^f. \tag{8.5}$$

Another interesting property of the DP PSFs is that the orientation of the "half CoC"

(a) All-in-focus input (b) Our synthetic bokeh

(c) All-in-focus input (d) Our synthetic bokeh

Figure 8.6: Our synthetic bokeh results given an input all-in-focus image. The images used in this figure are from the synthetic DoF dataset [27].

of each left/right view reveals if the scene point is in front or back of the focal plane [4, 110, 127]. Based on the DP image formation discussed in the previous chapters, we also select the DP-based "half CoC" PSF model to capture the directional blur in this paper. However, this directional blur PSF does not have to be DP-based and can be any generic

(a) Our synthetic DP views (b) Real DP views

Figure 8.7: Results from our DP-view synthesis framework based on defocus blur in DP sensors. (a) Our synthetic DP views. (b) Real DP views. Our framework can produce DP views that have defocus disparity similar to the one found in real DP sensors. The image on the left is from the synthetic DoF dataset [27]. **Note: the DP views are animated; click on the image to start the animation. It is recommended to open this PDF in Adobe Acrobat Reader to work properly.**

PSF that involves blurring and shifting the image content. Therefore, we test other non-DP-based directional PSF in Section 8.4.2.

8.3.3 Applying Synthetic Defocus Blur

In our framework, we use an estimated depth map to apply synthetic defocus blur in the process of generating a shallow DoF image. To blur an image based on the computed CoC radius r, we first decompose the image into discrete layers according to per-pixel depth values, where the maximum number of layers is set to 500 (similar to [68]). Then, we convolve each layer with the DP PSF (Section 8.4), blurring both the image and mask of the depth layer. Next, we compose the blurred layer images in order of back-to-front, using the blurred masks. For an all-in-focus input image $\mathbf{I}_s$, we generate two images – namely, the *left* $\mathbf{I}_l$ and *right* $\mathbf{I}_r$ sub-aperture DP views – as follows (for simplicity, let $\mathbf{I}_s$ be a patch with all pixels from the same depth layer):

$$\mathbf{I}_l = \mathbf{I}_s * \mathbf{H}_l, \tag{8.6}$$

$$\mathbf{I}_r = \mathbf{I}_s * \mathbf{H}_r, \tag{8.7}$$

where $*$ denotes the convolution operation. The final output image $\mathbf{I}_b$ (i.e., synthetic shallow DoF image) that is produced by the traditional *portrait mode* can be obtained as follows:

$$\mathbf{I}_b = \frac{\mathbf{I}_l + \mathbf{I}_r}{2}. \tag{8.8}$$

Figure 8.6 shows the results of the generated synthetic bokeh image $\mathbf{I}_b$ using our proposed framework. Furthermore, our synthetically generated DP views exhibit defocus disparity similar to what we find in real DP data, where the in-focus regions show no disparity and the out-of-focus regions have defocus disparity. We provide in Figure 8.7 an animated comparison between our generated DP views and real DP views extracted from a Canon DSLR camera.

8.3.4 Multi-View Synthesis

The main idea of this work is to generate multiple views from an all-in-focus image with its corresponding depth map. Therefore, we can generate an aesthetically realistic image motion by synthesizing a multi-view version of a given single image. As discussed in Section 8.3.2, the DP two sub-aperture views of the scene depending on the sensor's orientation and, in this work, our formation contain left/right DP pairs, and consequently, our framework synthesizes the horizontal DP disparity as shown in Figure 8.7. We can synthesize additional views with different "DP disparity" by rotating the PSFs during the multi-view synthesis process as shown in Figure 8.3. For example, eight views can be generated by performing a 45^o clockwise rotation step three times (i.e., 45^o, 90^o, 135^o). Then, we generate our effect by alternating the output views to produce the image motion.

8.4 Experiments

8.4.1 Results Using Dual-Pixel Blur Kernel

Following the qualitative comparison procedure introduced in [127], we provide the animated image motion (or NIMAT effect) of different approaches in Figure 8.8. In particular, we compare ours with the results from [127] and the Facebook 3D image. As mentioned earlier and unlike other approaches, our proposed framework starts with the deep DoF image (i.e., almost all-in-focus) to produce the synthetic bokeh (or synthetic shallow DoF) image and the multiple DoF/DP-based views. Therefore, we provide the synthetic bokeh

(a) Facebook 3D image (click) (b) NIMAT effect [127] (click) (c) Our NIMAT effect (click)

(d) Facebook 3D image (click) (e) NIMAT effect [127] (click) (f) Our NIMAT effect (click)

Figure 8.8: A comparison between different image motion approaches. This image motion is produced by animating the synthetic output views of each approach. Two cases of scene depth variation are provided: a small depth variation in the first row and a large one in the second row. Our proposed image motion produces a pleasant motion transition and fewer artifacts compared to others. The images used in this figure are from the synthetic DoF dataset [27]. **Note: the synthetic output views are animated; click on the image to start the animation. It is recommended to open this PDF in Adobe Acrobat Reader to work properly.**

(a) NIMAT – DP PSF (click) (b) NIMAT – Ramp PSF (click)

(c) DP PSF (d) Ramp PSF

Figure 8.9: A comparison between different PSFs used to render the NIMAT effect. The two PSFs (i.e., DP PSF and Ramp PSF) are able to render smooth image motion. However, different motion transitions and artifacts can be introduced by using different PSFs. **Note: the synthetic output views are animated; click on the image to start the animation. It is recommended to open this PDF in Adobe Acrobat Reader to work properly.**

image as input to other approaches. This section also introduces the NIMAT-like effect from the common Facebook 3D image by uploading a single image and rendering the 3D version. Then, we save multiple frames at different view directions following the circular pixel motion transition found in the NIMAT effect [127].

The results in this section show two cases of scene depth variations – namely, a small depth variation (Figure 8.8, first row) and a large one (Figure 8.8, second row). While the Facebook 3D image motion is sufficient in the first row, it suffers from few artifacts

around the foreground object boundary (e.g., the wall behind the person's head and arm). As for the NIMAT effect results from [127] in the first row, the image motion is barely noticeable in the background due to the small blur size that is a result of the small scene depth variation.

The second row of Figure 8.8 shows the large depth variation case, where the blur size varies from small to large. In this case, the Facebook 3D image exhibits noticeable and unpleasing artifacts (e.g., missing pixels). While the NIMAT effect from [127] produces pleasing image motion, we can still spot few artifacts that do not exist in ours. Note that we are aware the Facebook 3D image is not made for the same purpose, but we rendered it with the same motion transition settings of the NIMAT effect for comparison purposes.

8.4.2 Results Using Other Blur Kernels

As mentioned earlier in Section 8.3.2, the directional PSF used to render the NIMAT effect can be any generic PSF that involves blurring and shifting the image content. In Figure 8.9, we show the NIMAT effect rendered using two different PSF shapes – namely, DP-based PSF (Figure 8.9, c) and transitional blurring 2D ramp mask with a constant intensity fall-off towards the opposite direction (i.e., Ramp PSF in Figure 8.9, d). These results demonstrate that other non-DP-based PSF can be utilized to render the NIMAT effect as long as it satisfies the conditions of having a transnational and blurring operator. Nevertheless, different motion transitions and artifacts can be introduced by using different PSFs as shown in Figure 8.9.

8.5 Conclusion

In this chapter, we proposed a modification to the DoF synthesis associated with most smartphones' *portrait mode* feature. This modification can be easily integrated into the traditional DoF synthesis unit and enables the generation of multiple sub-aperture views along with the synthetic bokeh photo. With this modification, we are also able to produce an aesthetic image motion effect similar to the novel NIMAT effect from [127]. For our multi-view synthesis, we introduced the novel idea of convolving the input image with the rotated blurring kernels based on the DoF blur and DP image formation. We validated our approach qualitatively and demonstrated that it produces smooth motion transition

in the NIMAT effect with fewer artifacts compared to others. We aim to encourage work in this new research direction that presented a new pleasing effect of image motion.

Part IV

Conclusion and Future Work

Chapter 9

Conclusion and Future Work

9.1 Conclusion

This thesis focused on camera DoF manipulation with emphasis on the new DP sensor technology. In addition to preserving and/or restoring important image details, DoF can be adjusted for a better photography experience by synthesizing pleasing shallow DoF. This work also showed that manipulating camera DoF can assist in boosting the performance of other computer vision tasks (e.g., image segmentation and depth estimation). In particular, we addressed camera DoF manipulation using DP sensor from two point of views i.e., pre- and post-image capture in Part II and Part III, respectively.

Pre-image capture DoF manipulation Camera AF is the common application for manipulating DoF before image capture. Therefore, this thesis has revisited the traditional AF problem and reviewed conventional methods for AF used on cameras (Chapter 2). The need to revisit AF in the context of video capture was also discussed. As part of this effort, a new software platform and dataset focused on AF for video capture with cameras were introduced in Chapter 3. Specifically, we constructed a hardware setup that allows dynamic scenes to be accurately repeated. Using this environment, we analyzed the AF behaviour of representative smartphone cameras under ten different scenes with various motions, backgrounds, and objects (including an object serving as a proxy for a human face). We also captured these scenes with discrete time points, producing a 4D temporal focal stack dataset for use in AF research. The overall dataset consists of 33,000 smartphone camera images and is publicly available. We also developed an AF platform

that allows the development of AF algorithms within the context of a working camera system. API calls allow algorithms to simulate lens motion, image access, and low-level functionality, such as phase and contrast detection. This platform also restricts an AF algorithm to operate within a real camera environment, where lens motion that is directly tied to the systems clock cycle and scene motion is required to access different images in the focal stack.

From our analysis of the cameras' AF behaviour we examined four high-level AF objectives—namely, global, 9 focus points, 51 focus points, and face region. Using our AF platform, we implemented these high-level AF objectives to produce several video outputs that were used in a user study. Because our AF platform allowed accurate analysis of the underlying AF conventional methods, we were able to determine that user preference is correlated higher to the overall lens motion than the actual scene objective used. For scenes with faces, focusing on the face (when sufficiently large) took priority, followed by the amount of lens motion. While this may seem an intuitive finding (e.g., no one likes a scene with too much lens wobble), as far as we are aware, this is the first study to confirm these preferences in a controlled manner.

Based on our initial user study finding, we also proposed supervised and unsupervised methods to perform *online* lens smoothing for video AF (Chapter 4). Our supervised approach uses a Bi-directional LSTM and demonstrates that a learning framework can be trained with smoothed data generated in an *offline* manner, but then operates in an *online* manner. As for the unsupervised approach, we introduced our WMA based on a moving average. We showed that our WMA is effective, yet simple, and does not require any training. Additionally, it can be easily integrated to current AF systems as it is computationally inexpensive. Our quantitative and qualitative results showed that our two *online* smoothing methods (BLSTM and WMA) performed on par with *offline* smoothing and results in lens motion reduction (up to 64% reduction) with very small change in sharpness (less than 5.2%). We also performed another user study demonstrated that our two *online* smoothing methods were notably preferred over conventional video AF. To the best of our knowledge, this is the first work in the literature to examine lens smoothing for video AF. We believe having access to our temporal focal stack dataset, AF platform, our novel AF smoothing algorithms, and user study findings will be useful in spurring future work in this area.

Post-image capture DoF manipulation In this thesis, we discussed two applications related to DoF adjustments after effect, including, DoF extension (i.e., defocus deblurring in Chapters 5–7)) and shallow DoF synthesis (i.e., bokeh effect in Chapter 8). We proposed a data-driven DNN framework for defocus deblurring and demonstrated that DP data can be very useful for this task. Such a data-driven deblurring approach required us to capture a new image dataset consisting of blurred and sharp image pairs along with their DP images (Chapter 5). We also demonstrated that our deblurring method can be beneficial for other computer vision tasks. By showing the importance of reducing defocus blur and to encourage research in this direction, we organized an NTIRE challenge for defocus deblurring using DP data in conjunction with CVPR 2021 (details in Appendix A.5).

In Chapter 6, we explored a new application domain where the collection of DP ground truth data is not feasible (e.g., fixed aperture smartphones, video data). To this aim, we proposed a generalized model of the DP image acquisition process that allowed realistic generation of synthetic DP data using standard computer graphics-generated imagery. By enabling the generation of time series DP data and DP image sequence (e.g., video), we expanded our DNN framework to a new recurrent convolutional architecture that is designed to reduce defocus blur in image sequences. Our new recurrent convolutional framework involved novel modifications like our edge loss function, training with radial distance patches, and adding convLSTM units. We performed a comprehensive evaluation of existing deblurring methods, and demonstrated that our synthetically generated DP data and recurrent convolutional model achieve state-of-the-art results quantitatively and qualitatively. Furthermore, our proposed framework demonstrates the ability to generalize across different cameras by training on synthetic data only.

Later in Chapter 7, we addressed the limitation of accessing DP data on-board camera as most DP cameras do not allow easy access to the DP sensor's two sub-aperture views. With that said, we proposed a DNN framework that does not require DP data to be available at inference time, which enabled testing on a wider range of different cameras that employ DP imaging sensor. We also demonstrated that a DNN trained for the purpose of single-image defocus deblurring can be improved by incorporating the additional task of synthesizing the two DP views associated with the input image. One benefit of this approach is that capturing data for the DP view synthesis task is easy to perform and requires no special capture setup. This is contrasted with a conventional approach that requires careful capture of sharp/blurred image pairs for the deblurring task. This multi-

task strategy improved deblurring results by close to 1dB in terms of PSNR. As an added benefit, we showed that our DNN is able to perform realistic view synthesis that can be used for tasks such as reflection removal and generating aesthetically realistic image motion.

Lastly, in Chapter 8, we investigated another application of post-capture DoF adjustment through synthesising shallow DoF. For that purpose, we presented a hand-crafted method with a new modification on the traditional camera portrait mode framework. Our modification enabled better bokeh effect as well as multi-view synthesis based on DP image formation.

We believe the contributions made in this thesis including the datasets, DoF adjustment methods, realistic models based on DP image formation, benchmarks, and defocus deblurring challenge will be useful for the research community and will help spur additional ideas in advancing DoF manipulation and applications that can leverage data from DP sensors. All datasets, source codes, and trained models are made publicly available to enable reproducibility and facilitate development.

9.2 Future Research Directions

What to focus on! Which part of the scene is the most salient has always been an interesting AF question. Based on our finding in the first user study 3.4, with the exception of faces in the foreground, there was no conclusive finding to existing RoIs used on cameras (e.g., 9-point, 25-point, full-frame). This finding inspires us to start thinking of ways to predict the preferred RoI. One potential method would be to employ eye tracking-based AF, in which the RoI dynamically changes based on the user's eye gaze. With the current advancement of smartphone computational power and front-facing camera, eye tracking has been reliably applied on smartphones in [152] without the need for additional sensors or devices. Therefore, an interesting research direction is to utilize a front-facing camera to involve eye-tracking technology in the process of camera manual focus. The idea is to estimate the user's eye gaze and then change the camera focus to the RoI in real-time and during the live preview. This idea will replace the traditional manual focus where users need to hold their finger on the touchscreen in order to lock the focus to a certain region. This finger-holding manual focus is tedious especially when it comes to video (camera and scene are moving), where users need to keep manually chasing the object of interest. The

idea is to move beyond traditional camera interfaces. As far as we are aware, no such concept has been introduced before.

Another idea that can help in reducing the computational power of real-time eye-tracking process is to predict the human gaze by leveraging recent advances in deep learning space for processing and analyzing visual data. With that said, the eye tracking-based AF idea is expected to benefit from deep learning models. The eye tracker can be used in *offline* manner to find the human preference and collect the RoIs for each scene in our 4D temporal focal stack dataset (Chapter 3). Then, each image in the dataset will be labeled in order to train a deep model that can learn the human preference over time. Once a deep model is trained, it can be employed on-board camera to process new visual data that have never been seen before in order to predict human AF preference.

RoI tracking in video AF In addition to determining the RoI for an AF algorithm, AF-based tracking is another interesting topic that involves object motion detection and tracking. One can think of motion trajectory prediction in the context of dynamic scenes for video AF. Such RoI tracking and prediction can also provide good cues to optimize AF lens motion accordingly.

In cinematography, there is a crew member called a focus puller who is primarily responsible for maintaining the camera lens focus on the subject or action that is being filmed. With that said, it is very useful to have a reliable video AF system that can detect and track the object of interest, optimize lens focus, and perform or assist the focus puller task for better cinematography.

Unsupervised defocus deblurring While we provided an unsupervised method for *online* lens motion smoothing (i.e., WMA in Section 4.5), we adopted a supervised data-driven approach for all our defocus deblurring methods. Unlike the supervised approach, the unsupervised one has many advantages like it can be less complex and easily integrated into new camera systems with a need to retrain. The most important advantage, that we focus on, is the need for annotated data as the unsupervised approach works with datasets without labeled ground truth.

In addition to the imperfections (e.g., local misalignment) that come with the capturing procedure, Chapter 6 and Chapter 7 clearly stated the difficulty behind collecting ground truth all-in-focus images. To this end, adapting an unsupervised method, to reduce defocus

blur, can help in avoiding inherited imperfections accompanying the capturing procedure with minimal data collection effort. The idea is mainly to leverage the symmetric property of PSFs found in DP sensors (Chapter 6). One can think of designing an optimization method with a hard constraint to guarantee such PSF symmetry to determine the PSF size at each pixel (something similar to our DP modeling in Section B.4.1 of Appendix B). Then, a non-blind deconvolution method can be applied by considering realistic defocus blur factors such as radial distortion, optical aberration, and the spatially varying nature of the PSF shape.

DoF manipulation software Another promising idea is to leverage recent advances in machine learning to come up with a photo editing software that enables manipulating camera DoF after effect. Such post-capture editing tool can be very useful for camera consumers, especially for the smartphone camera consumer as these cameras are more challenging given their simple imaging systems and optics. This tool can facilitate several after-effect editing such as shifting DoF (e.g., refocusing in Chapter 3), extending DoF (e.g., defocus deblurring in Chapter 5), and mimicking shallow DoF (e.g., synthetic bokeh in Chapter 7).

Others It is always useful to study the data-driven and learning-based algorithms in order to understand what they are exactly doing. Investigating these algorithms thoroughly is the first step towards optimizing and improving them.

Appendices

Appendix A

Defocus Deblurring Challenge

As mentioned in Chapter 5, reducing defocus blur is as important as reducing motion blur. However, defocus deblurring receives much lower attention compared to motion deblurrig. To encourage research on the defocus deblurring area, we organized this NTIRE challenge in conjunction with CVPR 2021 [109].

This Appendix provides a review of the NTIRE 2021 challenge targeting defocus deblurring using dual-pixel (DP) data. The goal of this single-track challenge was to reduce spatially varying defocus blur present in images captured with a shallow depth of field. The images used in this challenge were obtained using a DP sensor that provided a pair of DP views per captured image. Submitted solutions were evaluated using conventional signal processing metrics, namely peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM). Out of 185 registered participants, nine teams provided methods and competed in the final stage. The chapter describes the methods proposed by the participating teams and their results. The winning teams represent the state-of-the-art in terms of defocus deblurring using DP images.

A.1 Introduction

Defocus blur occurs in an image at scene points when the light rays, traveling through the camera optics, converge either before or after the imaging sensor. While the effect of defocus blur can be intentional in photography (e.g., the bokeh effect [27, 139]), for computer vision applications, defocus blur is often undesired and affects image quality due to the loss of sharp image details. Recovering sharper details from a defocus deblurred

image is challenging [71] due to the spatially varying nature of blur shape and size because of optical aberrations across scene depth [72].

In Chapter 5, we demonstrated the advantage of utilizing dual-pixel (DP) data to reduce defocus blur. Therefore, this challenge is designed to reduce image DoF blur given its DP views. This challenge is one of the NTIRE 2021 associated challenges: nonhomogeneous dehazing [153], defocus deblurring using dual-pixel [109], depth guided image relighting [154], image deblurring [155], multi-modal aerial view imagery classification [156], learning the super-resolution space [157], quality enhancement of heavily compressed videos [158], video super-resolution [159], perceptual image quality assessment [160], burst super-resolution [161], high dynamic range [162]. This single-track challenge solicited algorithms that can reduce defocus blur using DP data. The challenge, dataset, and the results based on PSNR and SSIM are described in the subsequent sections.

A.2 The Challenge

The NTIRE 2021 challenge on defocus deblurring using DP images is aimed to gauge and advance the state-of-the-art in reducing defocus blur. This challenge aims to evaluate the performance of the proposed defocus-deblurring methods on a newly captured DP dataset. The following provides a detailed description of the DP datasets used, the evaluation procedure and metrics, and the challenge timeline.

A.2.1 Datasets

Following the capturing procedure provided in Section 5.3, the dataset used in this challenge consists of images carefully captured on a tripod with the aperture adjusted between image captures. Two images are captured for the same static scene in succession; the first is a wide-aperture image captured using f/4 and exhibits notable defocus blur, while the second is captured with a narrow aperture (i.e., f/22) and exhibits a wide depth of field with almost no defocus blur. The narrow aperture image serves as the ground truth sharp image.

The training data used for this challenge consists of the 500 indoor/outdoor scenes from the DP defocus deblurring dataset[4] [3] (Chapter 5). In addition to this dataset, we

[4] `https://abuolaim.nowaty.com/eccv_2020_dp_defocus_deblurring/`

captured a new 100 indoor/outdoor scenes divided equally for the validation and testing data. The challenge provides 600 scenes in total (i.e., 1800 images) where each scene has: (i) the two DP sub-aperture views of an image with defocus blur captured at a large aperture; and (ii) the corresponding all-in-focus image captured with a small aperture.

The dataset images are high-quality as they are captured with low ISO (i.e., low ISO equates to low-noise [94]) and have a $1,680 \times 1,120$ spatial resolution. These images, including the left/right DP views, are processed to an sRGB color space and encoded with a lossless 12-bit depth per RGB channel.

A.2.2 Evaluation

The evaluation compares the recovered sharp (i.e., deblurred) images with the ground-truth images. For this comparison, we use the standard peak signal-to-noise ratio (PSNR) and the structural similarity (SSIM) index [103] as often employed in the literature. We report the average results over all the estimated test images provided.

Challenge participants were asked to provide the estimated deblurred images with the same resolution as the input and using the specified naming convention. Participants were also asked to provide additional information, for example, the algorithm's runtime per test image (in seconds); whether the algorithm employs CPU or GPU at runtime, and whether extra metadata is used as inputs to the algorithm.

At the final stage of the challenge, the participants were asked to submit fact sheets to provide information about the teams and describe their methods. In addition to the fact sheets, the output results, codes and trained models were also submitted.

A.2.3 Timeline

The challenge timeline had two stages – validation and testing. The validation stage commenced on January 6, 2021, and lasted for approximately 10 weeks. The final testing stage started on March 15, 2021, and lasted for 5 days. Each participant was allowed 20 submissions during the validation stage and three submissions for the testing phase. The challenge ended on March 20, 2021. The final test results were shared with the participants on March 26, 2021.

Table A.1: Results and rankings by team of the submitted methods for NTIRE 2021 defocus deblurring challenge. MS-SSIM [163]: multi-scale structural similarity loss. The runtime of s/Mpixel is calculated based on the runtime of a single test image of size $1680 \times 1120 \times 3$ pixels (i.e., 5.6448 Mpixel).

Team	Username	PSNR	SSIM	Runtime (s/Mpixel)	CPU/GPU (at runtime)	Platform	Ensemble	Loss
SRCB	SRCB-Z	$27.80_{(1)}$	$0.8484_{(1)}$	$0.029_{(1)}$	Tesla V100	PyTorch [91]	flip, multi-model ensemble	Charbonnier
TeamInception	swz30	$27.48_{(2)}$	$0.8387_{(3)}$	$0.514_{(7)}$	Tesla V100	PyTorch [91]	flip, rotate, self-ensemble [164]	L_1, VGG [165], MS-SSIM [163]
Mier	q935970314	$27.13_{(3)}$	$0.8408_{(2)}$	$10.62_{(9)}$	Tesla V100	PyTorch [91]	left/right swap, self-ensemble [164]	L_1
VIDAR	zyxiao	$26.79_{(4)}$	$0.8264_{(4)}$	$0.046_{(3)}$	Tesla V100	PyTorch [91]	flip, inverse transform	Charbonnier, SSIM
Attention	buffalo	$26.42_{(5)}$	$0.8020_{(6)}$	$0.097_{(5)}$	Tesla V100	Tensorflow–Keras [101]	flip/rotate	L_2, SSIM
Maradona	hellosr	$26.38_{(6)}$	$0.8017_{(7)}$	$08.70_{(8)}$	Tesla V100	PyTorch [91]	Geometric self-ensemble×8 [166]	L_1
AIIA	huxingyu	$26.15_{(7)}$	$0.8021_{(5)}$	$0.482_{(6)}$	RTX 3090	PyTorch [91]	None	L_2
DDDP	BaiXiaoying	$25.17_{(8)}$	$0.7620_{(8)}$	$0.081_{(4)}$	TITAN Xp	PyTorch [91]	None	L_2, SSIM
ImageLab	sabarinathan	$24.85_{(9)}$	$0.7390_{(9)}$	$0.041_{(2)}$	1080 GTX	Tensorflow–Keras [101]	None	L_2, SSIM, Sobel

A.3 Results

Out of 185 registered participants, the challenge had nine teams who continued to the final stage by submitting results, codes/executables, and fact sheets. Tables A.1 reports the final test results based on PSNR and SSIM index [103]. The overall method rank

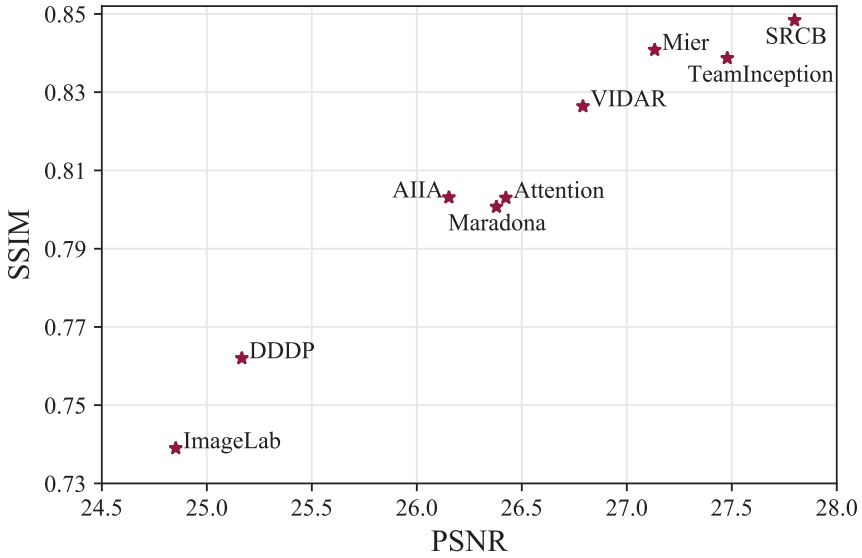


Figure A.1: A 2D visualization of the combined PSNR and SSIM values of the proposed methods. Each team name is shown next to the scattered star markers.

based on each metric is indicated in subscripts. Additionally, the self-reported runtimes and major details, provided in the fact sheets submitted by participants, are also reported in Table A.1. Fig A.1 show a 2D visualization of PSNR and SSIM values for all teams. The team members along with affiliations are listed in Section A.5.

A.3.1 Core Idea

All of the proposed methods are based on deep learning. Particularly, all methods employ different architectures of deep convolutional neural networks (CNN). Most of proposed architectures are based on widely used CNN-based networks, such as Siamese network [167], U-Net [96] and ResNet [97]. The core ideas included re-structuring existing networks, introducing skip connections, introducing residual connections, and using densely connected components. Other strategies have been utilized including supervised attention module (SAM) [168], residual channel attention blocks [169], dual-attention modules [170], convolutional block attention module (CBAM) [171], and dilation blocks [19, 172].

As for loss functions, different types were employed such as L_1, L_2, Charbonnier, SSIM, multi-scale SSIM [163] (MSSSIM), VGG [165], and Sobel. Some teams used a single loss, whereas others used a mix of loss functions (e.g., TeamInception utilized L_1, VGG [165],

and MSSSIM [163]).

A.3.2 Top Results

The SRCB team has achieved the best results for all metrics, including the inference time (i.e., 0.029 s/Mpixel). They are 0.32 dB higher compared to the second top team i.e., TeamInception as reported in Table A.1. In terms of PSNR, the main performance metric used in the challenge, the top three methods achieved larger than 27dB, and they are from the teams of SRCB, TeamInception, and Mier, respectively (see Figure A.1). In terms of SSIM, as a complementary performance metric, the second-best method is proposed by the Mier team and achieved a SSIM index of 0.8408, while the third-best SSIM index is TeamInception team, i.e., 0.8387.

A.3.3 Ensembles

Most of the teams applied different flavors of ensemble techniques to boost the overall performance. In particular, most teams used a self-ensemble [164] technique where the results from flipped/rotated versions of the same image are averaged together. Some teams applied additional techniques such as inverse transform, geometric self-ensemble [166], and multi-model ensemble.

A.4 Conclusion

In this appendix, we evaluated the methods submitted for the NTIRE 2021 challenge for defocus deblurring using DP images with focus on their results. In particular, an evaluation of methods proposed by nine teams is performed based on PSNR, SSIM, and inference time. In addition to the DP deblurring dataset from [3] (Chapter 5), we provided a new DP-based deblurring dataset for further evaluations. All the proposed methods are based on deep CNNs and most of them utilized attention layers to boost performance. The best performing method is proposed by the SRCB team from Samsung research Beijing, China, where they achieved the best results for all metrics. This NTIRE challenge was organized in conjunction with CVPR 2021 [109].

Related links:

- Challenge website and evaluation server:
  `https://competitions.codalab.org/competitions/28049`
- CVPRW presentation:
  `https://www.youtube.com/watch?v=OC52DLyz1lU`

A.5 Team Names and Affiliations

NTIRE 2021 Organization

Title: NTIRE 2021 Challenge for Defocus Deblurring Using Dual-pixel Images: Methods and Results

Members:

Abdullah Abuolaim[1] (abuolaim@eecs.yorku.ca),
Radu Timofte[2] (radu.timofte@vision.ee.ethz.ch),
Michael S. Brown[1] (mbrown@eecs.yorku.ca)

Affiliations:

[1] York University, Canada
[2] ETH Zurich, Switzerland

Samsung Research China – Beijing (SRC-B)

Title: MRNet: Multi-Refinement Network for Dual-pixel Images Defocus Deblurring

Members:

Dafeng Zhang (dfeng.zhang@samsung.com), Xiaobing Wang

Affiliations:

Samsung Research China – Beijing (SRC-B), China

TeamInception

Title: Multi-Stage Progressive Image Restoration

Members:

Syed Waqas Zamir[1] (waqas.zamir@inceptioniai.org), Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ling Shao

Affiliations:

[1] Inception Institute of Artificial Intelligence (IIAI), UAE

Mier

Title: Big UNet

Members:

Shuai Liu[1] (18601200232@163.com), Lei Lei[2] , Chaoyu Feng[2]

Affiliations:

[1] North China University of Technology, China

[2] Xiaomi, China

VIDAR

Title: Multi-Scale Feature Fusion Network for Defocus Deblurring Using Dual-Pixel Images

Members:

Zhiwei Xiong (zwxiong@ustc.edu.cn), Zeyu Xiao, Ruikang Xu, Yunan Zhu, Dong Liu

Affiliations:

University of Science and Technology of China, China

Attention

Title: Attention! Stay Focus!

Members:

Tu Vo (tuvovan@pukyong.ac.kr)

Affiliations:

Bridge AI Inc., Korea

Maradona

Title: Multi-scale CNN for Dual Defocus Deblurring

Members:

Si Miao[1] (miaosi2018@sari.ac.cn), Nisarg A. Shah[2]

Affiliations:

[1]Shanghai Advanced Research Institute, Chinese Academy of Sciences, China

[2]Indian Institute of Technology Jodhpur, India

AIIA

Title: Multi-Branch Convolution Neural Network for Dual-Pixel Image Deblurring

Members:

Pengwei Liang[1] (erfect@stu.hit.edu.cn), Zhiwei Zhong, Xingyu Hu

Affiliations:

[1]Harbin Institute of Technology, China

DDDP

Title: Defocus Deblurring via Dual-pixel Images

Members:

Yiqun Chen[1] (chenyiqun2021@ia.ac.cn), Chenghua Li, Xiaoying Bai, Chi Zhang, Yiheng Yao, Ruipeng Gang

Affiliations:

[1] Institute of Automation, Chinese Academy of Sciences, China

ImageLab

Title: A Dual Vision Net : A novel approach for Deblurring the Dual-pixel Images

Members:

Sabari Nathan[1] (sabarinathantce@gmail.com), Thangavelu Ragavendran[2], Venkatakrishnan Srinija[3], Venkatakrishnan Srivatsav[4]

Affiliations:

[1]Couger Inc., India

[2]Jeppair College of Engineering, India

[3]Sri Sai Ram Institute of Technology, India

[4]Prince Shri Venkateshwara Padmavathy Engineering College, India

Appendix B

Modeling defocus disparity

In Chapter 6, we proposed a new modeling of the PSFs found in DP sensors. For that DP-PSF modeling, we thoroughly modeled the complete DP image acquisition process including a spatially varying PSFs that accounts for radial lens distortion, optical aberration, and image noise. In this appendix, we introduce a simplified DP-PSF modeling that can serve as an alternative not only for reducing defocus blur but also for other tasks like depth estimation.

Recently, the DP views have been used for tasks beyond autofocus, such as synthetic bokeh, reflection removal, and depth estimation. These recent methods treat the two DP views as stereo image pairs and apply stereo matching algorithms to compute local disparity. However, dual-pixel disparity is not caused by view parallax as in stereo (as explained in Chapter 6), but instead is attributed to defocus blur that occurs in out-of-focus regions in the image. This appendix proposes a new, yet simple, parametric PSF to model the defocus-disparity that occurs on DP sensors. We apply our model to the task of depth estimation from DP data. An important feature of our model is its ability to exploit the symmetry property of the DP blur kernels at each pixel. We leverage this symmetry property to formulate an *unsupervised* loss function that does not require ground truth depth. We demonstrate our method's effectiveness on both DSLR and smartphone DP data. The work presented in this appendix is a part of our publication in the International Conference on Computational Photography (ICCP), 2020 [4].

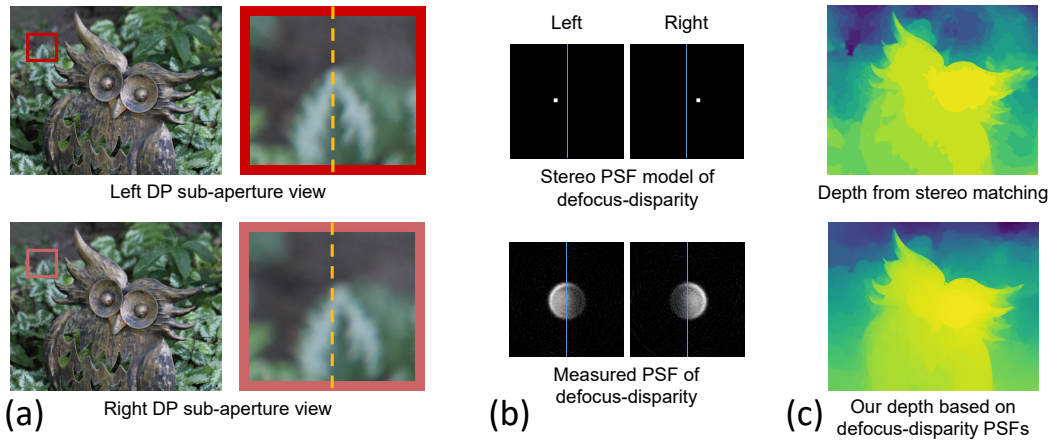


Figure B.1: (a) Shows the two sub-aperture views (denoted as *left* and *right*) from a DP sensor. An image region is zoomed in. The dotted reference line illustrates the disparity between the two views. (b) Shows the shifted impulse kernels for classic stereo and the empirically measured defocus kernels corresponding to the center pixels of the left and right zoomed-in DP patches. Classic stereo matching attempts to map one image to the other through an explicit shift in image content. This shift can be modeled as an impulse PSF. Disparity in a DP sensor, however, is the result of defocus blurring by two different PSFs acting in opposing directions. (c) Our method, which models the disparity in a DP sensor using defocus blur kernels, is able to estimate a more accurate depth map than algorithms based on traditional stereo matching (e.g., [27]). Note that the depth map is slightly blurry due to our new PSF and the post-processing edge-aware filtering applied.

B.1 Introduction

Dual-pixel (DP) sensors use a unique design that splits each pixel site in half using two photodiodes. This split-pixel arrangement acts as a rudimentary light-field camera and allows the sensor to capture *two* sub-aperture views of the scene in a *single* exposure. Image regions that are outside the optical depth of field will observe a disparity between the two DP sub-aperture views. Camera systems use this information to determine how to move the lens in order to minimize the dual-pixel disparity in specific regions of interest, and thereby bring those regions into focus. While the DP sensor was designed for the purpose of autofocus, many recent papers have used these two DP views for other tasks, such as enhanced synthetic bokeh [27], reflection removal [173], and depth reconstruction [25].

These methods treat the DP images as stereo image pairs and estimate disparity using stereo matching techniques.

Despite the resemblance to classic stereo, the nature of the disparity in DP sensors has key differences from stereo disparity. Classic stereo treats disparity as an explicit shift in image content between the two images. This can be modeled as a PSF with a single impulse as shown in Figure B.1. The split-pixel arrangement in a DP sensor in conjunction with the camera optics has the effect that light rays passing through the right side of the lens are captured by the left half-pixels (left sub-aperture view) and those passing through the left side of the lens are collected by the right half-pixels (right sub-aperture view). For an in-focus region of the scene, there will be no disparity between the left and right DP views. However, for a region of the scene that is away from the focal plane, a *defocus-disparity* is induced by the out-of-focus blur being split across the two views in opposing directions (see Figure B.1(a)). Thus, unlike in stereo, it is the difference in the point spread functions (PSF) that produces disparity between DP views and not an explicit shift in image content. This is illustrated in Figure B.1(b) that shows the empirically measured PSFs for the left and right DP views.

This appendix examines the nature of defocus-disparity in DP sensors and proposes a parameterizable PSF to model the blur kernels based on empirical observations of both DSLR and smartphone DP data. A useful property of this model is that the PSF in one DP view equals the PSF in the other DP view flipped about the axis perpendicular to the disparity axis. Using this symmetry property, we describe how to formulate an unsupervised loss function for the task of depth estimation from DP data. Our unsupervised loss has the advantage of using only data from the DP sensor, and circumvents the need for ground truth depth which is hard to acquire. Moreover, by explicitly formulating DP disparity as a defocus kernel in line with the image formation, our method makes judicious use of the depth cues embedded within the defocus blur. We establish the effectiveness of our loss function using a straightforward optimization-based approach. We demonstrate experimentally that our unsupervised optimization produces more accurate results than classic stereo matching algorithms (see Figure B.1(c)) and also compares favourably against state-of-the-art monocular and stereo-based deep learning methods for depth estimation that require supervised training. Additionally, we show that considerable speed-up can be achieved by replacing our optimization algorithm with a convolutional neural network (CNN).

Finally, we note DP data has an inherent ambiguity in that depth can be estimated only up to an unknown affine transformation [25]. This ambiguity is related to the optics of a DP sensor, and is similar to the well-understood scale ambiguity in depth from stereo (if the extrinsics are unknown). We refer readers to [25] for a detailed discussion on the connection between disparity and affine-transformed depth. Following [25], in this work, we also estimate depth up to an affine ambiguity.

B.2 Background

Early work on depth estimation relied on stereo pairs [174] or multi-view geometry [175]. These methods attempt to constrain the solution by assuming that multiple views of the scene of interest are available. Estimating depth from a single image is more ill-posed as the same input image can project to multiple plausible depths. Classic monocular depth estimation methods leveraged cues such as shading [176], contours [177], and texture [178] to infer depth. However, these early methods could be applied only under certain constrained scenarios. Following the seminal monocular depth estimation work of Saxena *et al.* [179], several approaches using hand-crafted features have been proposed [180–187].

The advent of deep learning saw rapid advancements in the area of monocular depth estimation. This was made possible mainly by end-to-end supervised training [188–192] on RGBD (RGB depth) datasets. Given the challenges in acquiring large amounts of ground truth depth in varied real-world settings, more recent work has explored the possibility of using synthetic depth data [124, 193–196] or using self-supervision for training [108, 140, 197–200]. The idea of self-supervision is to use stereo pairs or video sequences, and train the model to predict per-pixel disparities that map the input image to its nearby view.

Depth from defocus (DFD) [201] is another technique by which the depth of the scene can be recovered. Here, depth is estimated from a stack of images obtained by varying the camera's focus [82, 85]. Levin *et al.* [105] showed that replacing a conventional camera's aperture with a coded aperture enables depth recovery from a single image capture by better exploiting focus cues. Our method too requires a single image capture, and works by explicitly modeling the relation between depth and defocus on a DP sensor using a parameterized PSF.

Defocus blur can act as a complementary cue to disparity for stereo matching [202]. Existing work has incorporated defocus cues into stereo disparity estimation. Rajagopalan

et al. [203] combine DFD and stereo matching using a Markov random field-based approach for robust depth estimation. However, their method requires two focal stack images from each stereo view. Bhavsar and Rajagopalan [204] generalize this framework to couple motion, blur, and depth. Methods for disparity estimation from a single pair of defocus stereo images have also been proposed [205, 206]. In comparison, the depth estimation algorithm of Paramanand and Rajagopalan [207] uses a blurred-unblurred image pair. To our knowledge, ours is the first work to explicitly model defocus blur for DP depth estimation. We exploit a symmetry property of the defocus kernels that is not present in a stereo configuration.

Depth can also be estimated using light field cameras [208, 209]. Light field cameras sample the 4D plenoptic function [210] using standard 2D images, and in so doing sacrifice spatial resolution for angular resolution. Due to their low spatial resolution, these cameras have not seen widespread consumer acceptance. A DP camera is also a rudimentary light field. However, the loss of spatial resolution in a DP sensor is minimal – only two angles from the light field are sampled, while light field cameras such as Lytro Illum sample 196 angles at the expense of considerable spatial resolution. Given their utility, this loss of spatial resolution is an acceptable compromise, and as such, DP sensors have been widely adopted on consumer cameras.

Recent work [25, 27] has explored the idea of depth estimation from DP sensor data. Wadhwa *et al.* [27] re-engineered classic stereo matching to estimate the depth map given the two DP views. The *folded loss* in the recently proposed deep learning method of Garg *et al.* [25], while having the advantage of being unsupervised, also models DP disparity as a simple per-pixel shift. This can lead to errors especially in regions far from the focal plane where the effective defocus PSFs vary vastly between the two views. Garg *et al.* [25] noted that their unsupervised loss is inadequate for the task and therefore propose a 3D supervised loss that assumes that the ground truth depth map is available during training. The requirement for ground truth depth poses additional challenges in terms of data acquisition, camera calibration and synchronization. Garg *et al.* [25] designed a custom-made calibrated rig containing five cameras, carefully synchronized capture, and used stereo techniques to estimate the "ground truth" depth map from the five views. We present an unsupervised loss function for depth estimation from DP data that does not require ground truth depth. Our method exploits an inherent symmetry of DP defocus kernels.

B.3 Dual-pixel Image Formation and Kernel Symmetry

In this section, we first examine the relationship between the PSFs corresponding to the left/right DP sub-aperture views and the image for an ideal DP sensor. Based on the image formation model in a DP camera, we describe a useful symmetry property of the left and right DP kernels. Next, we conduct a set of experiments on real data captured using DP cameras to study how the shape of the PSFs deviates in practice from the ideal case. We validate that the symmetry property holds equally well for real DP data.

A DP sensor splits a single pixel in half using two photodiodes housed beneath a microlens as shown in Figure B.2. The left and right photodiodes can detect light and record signals independently. The pixel intensity that is produced when these two signals are summed together is recorded as the final image, and will match the value from a traditional non-DP sensor. Also observe from Figure B.2 that left half-pixels integrate light over the right half of the aperture, and vice versa. We note that this can also be the upper and lower halves of the aperture depending on the sensor's orientation. Without loss of generality, we consider them to be the left and right views in the rest of the appendix.

Let us now examine in detail the image formation in an *ideal* DP camera as illustrated in Figure B.2. Consider a scene point that is at the focal plane of the lens (see Figure B.2(a)). Light rays emanating from this point travel through the camera lens and are focused at a single pixel. There is no disparity and the blur kernel is an impulse in the left and right DP views as well as the image. Next, consider the scene points in Figs. B.2(b) and (c) that are away from the focal plane. Light rays originating from the scene point in Figure B.2(b) that is behind the focal plane converge at a point in front of the sensor and produce a seven-pixel wide blur on the sensor. The circle of confusion that is induced by this out-of-focus scene point is split over the two DP views producing *half-circle* PSFs as shown. The sum of these two PSFs is equal to the full circle of confusion observed on the image since together, they account for all the light passing through the aperture. In a similar manner, rays from the scene point in Figure B.2(c) that is in front of the focal plane also create a seven-pixel wide blur on the sensor, and the left and right kernels are once again half-circles. While the combined kernel corresponding to the image is exactly the same as in Figure B.2(b), observe that the kernels corresponding to the left and right DP views have swapped positions, making it possible to disambiguate whether the scene point is behind or in front of the focal plane. Thus, the disparity is proportional to the

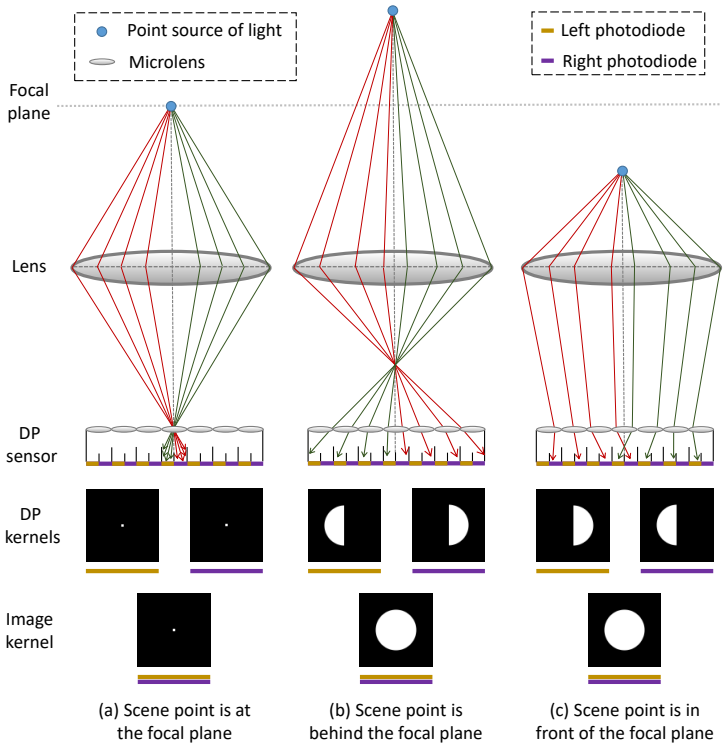


Figure B.2: An illustration of the image formation on an ideal DP camera. (a) A scene point that is at the focal plane produces no disparity – the PSF is an impulse in the left and right DP sub-aperture views as well as the image. (b-c) On the other hand, scene points that are away from the focal plane induce a *defocus-disparity* between the left and right DP views. For an ideal sensor and lens, the circle of confusion resulting from an out-of-focus scene point will be split across the two views leading to *half-circle* PSFs in the left and right views. The disparity is proportional to the blur circle radius, and the *sign* of the disparity can be determined from the direction of the half-circle PSFs, making it possible to disambiguate whether the scene point is behind or in front of the focal plane.

(signed) blur circle radius. The blur circle radius itself is a function of the diameter of the aperture, the focal length of the lens, the focus distance of the camera, and the scene depth [25].

Consider an image patch $\mathbf{G}$ corresponding to a constant-depth region in the scene. Let $\mathbf{G}_l$ and $\mathbf{G}_r$ denote its corresponding left and right DP views, respectively, and $\mathbf{F}$ denote

the underlying sharp patch (which is usually unknown). From the DP image formation model, we can write

$$\mathbf{G}_l = \mathbf{F} * \mathbf{H}_l, \tag{B.1}$$

$$\mathbf{G}_r = \mathbf{F} * \mathbf{H}_r, \tag{B.2}$$

$$\mathbf{G} = \mathbf{G}_l + \mathbf{G}_r = \mathbf{F} * (\mathbf{H}_l + \mathbf{H}_r) = \mathbf{F} * \mathbf{H}, \tag{B.3}$$

where $*$ denotes the convolution operation, $\mathbf{H}_l$ and $\mathbf{H}_r$ represent the PSFs that produce $\mathbf{G}_l$ and $\mathbf{G}_r$ when convolved with $\mathbf{F}$, and $\mathbf{H}$, which is the sum of $\mathbf{H}_l$ and $\mathbf{H}_r$, represents the combined kernel corresponding to the image $\mathbf{G}$.

Figure B.2 also reveals an interesting property of the kernels corresponding to the left and right DP views – the left kernel equals the right kernel flipped about the vertical axis (which is the axis perpendicular to the axis of disparity), and vice versa. Mathematically, we can express this as $\mathbf{H}_l = \mathbf{H}_r^f$, where $\mathbf{H}_r^f$ represents the right kernel flipped about the vertical axis. As we shall demonstrate in Section B.4, this symmetry property of the kernels plays a central role in the construction of our unsupervised loss function.

B.3.1 Kernels on a Real Dual-Pixel Sensor

The half-circle kernels shown in Figure B.2 correspond to an *ideal* sensor and lens. On a real sensor, due to physical constraints in the positioning of the microlens, depth of the sensor wells, and other manufacturing limitations, it is to be expected that a part of the light ray bundle passing through the left half of the lens will leak into the left-half dual pixels, and vice versa. To investigate how the shape of the kernels deviates from the ideal theoretical case and whether the symmetry of the PSFs still holds, we perform the following experiment on real data captured using DP cameras.

We capture images of calibration patterns, and then estimate the "ground truth" kernels corresponding to the image as well as the left/right DP views using a *non-blind* kernel estimation technique (since the underlying sharp images are known from the calibration patterns). Towards this goal, following [211], we use a grid of disks with a known radius and spacing as the calibration image (see Figure B.3(a)). The calibration pattern is displayed on a 27-inch LED display of resolution 1920×1080. Following [173], we use the Canon EOS 5D Mark IV DSLR camera, which provides access to the sensor's DP data, to capture images. The monitor is placed at a fixed distance of about one meter

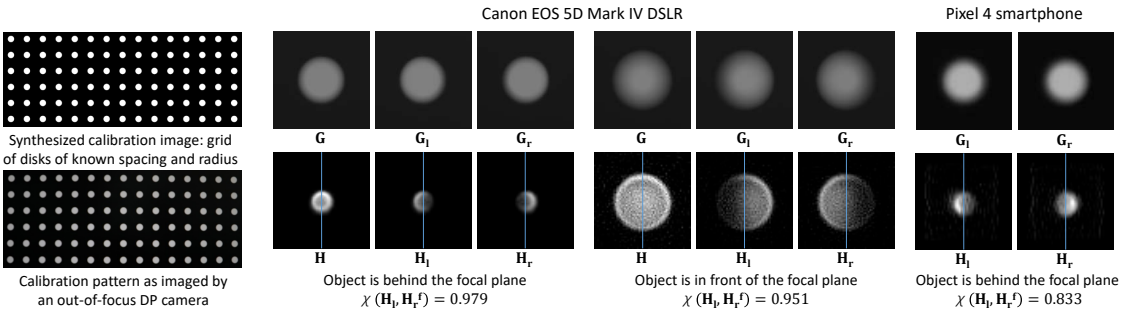


Figure B.3: We estimate "ground truth" kernels corresponding to the image as well as the left and right DP views independently. Using a grid of disks calibration pattern with known radius and spacing, we compute the sharp disk for non-blind kernel estimation. In the second and third columns, patches corresponding to the image, and the left and right DP views are shown for two different focus settings for data captured using the Canon EOS 5D Mark IV DSLR camera. Their associated estimated kernels are also shown in the second row. Notice that the direction of decay of the DP kernels is reversed depending on whether the object is behind or in front of the focal plane. The last column shows left and right DP patches and their corresponding estimated kernels for data captured using the Pixel 4 smartphone. The kernels are of size 61×61 pixels. The normalized 2D cross-correlation value χ between $\mathbf{H}_l$ and $\mathbf{H}_r^f$ is close to unity, indicating that our kernel symmetry property holds true in practice.

fronto-parallel to the camera, and the focus distance is varied such that the focal plane is either behind or in front of the monitor introducing different levels of defocus blur. To avoid radial distortion effects at the periphery, we use only 20–30 disks at the center of the image for blur kernel estimation. Patches containing these disks are identified, the center of the disks estimated by finding the centroid of these patches, and the disk patches averaged. We round the estimated centroid coordinates to the nearest integer pixel value to avoid resampling the disks during averaging. The radius of the disks is a known fraction of the distance between disk centers. Thus, following [211], the latent sharp disk image for non-blind kernel estimation can be generated based on the size of the disk grid in the captured image. To estimate the kernel $\mathbf{K}$ from a sharp-blurred image pair $(\mathbf{F}, \mathbf{B})$, we

solve the following equation:

$$\arg\min_{\mathbf{K}} \sum_{p} ||\mathbf{D}_p(\mathbf{F} * \mathbf{K} - \mathbf{B})||_2^2 + \gamma||\mathbf{K}||_1, \quad \text{subject to } \mathbf{K} \geq \mathbf{0}, \tag{B.4}$$

where $\mathbf{D}_p \in \{\mathbf{D}_\delta, \mathbf{D}_x, \mathbf{D}_y, \mathbf{D}_{xx}, \mathbf{D}_{yy}, \mathbf{D}_{xy}\}$ denotes the spatial derivatives along the horizontal and vertical axes, $\mathbf{D}_\delta$ is a zero-order gradient (i.e., an identity matrix), and γ is a positive scalar which we set to 1. The l_1-norm encourages the kernel entries to be sparse. Additionally, we impose non-negativity constraint on the entries of $\mathbf{K}$. Note that we do not explicitly enforce that $\mathbf{F}$ and $\mathbf{B}$ have the same average intensity, and so we do not impose the sum to unity constraint while solving for $\mathbf{K}$.

We extract the DP views from the captured data, and estimate kernels *independently* for the image as well as the left and right views. The second and third columns of Figure B.3 show patches and estimated kernels corresponding to the image and the left and right DP views at two different focus distances. As expected, the shape of the DP kernels deviates from the ideal half-circles; the fall-off is gradual. To quantatively evaluate whether the symmetry property holds for real data, we compute normalized 2D cross-correlation χ (which is a commonly-employed metric for kernel similarity [212]) between the left kernel $\mathbf{H}_l$ and the flipped version of the right kernel $\mathbf{H}_r^f$. A value of χ close to one means that the kernels are a close match. As seen from the χ values in Figure B.3, our kernel symmetry assumption is valid even on real data. This can be attributed to sensor imperfections likely affecting both left and right halves in the same manner. We computed the value of χ by varying the focus distance, the aperture, and the focal length, and obtained consistent results.

While DP sensors are used by most modern consumer cameras, DP data is not accessible to users on the vast majority of these devices. This is because after the autofocus operation (which is one of the very early stages of the camera pipeline), the split-pixel data is combined to produce the image.

To our knowledge, the Canon EOS 5D Mark IV is one of the few commercially available cameras in the DSLR segment that provide access to the sensor's DP data. Recently, the authors of [25] have released an Android application that allows DP data to be read from the Pixel 3 and Pixel 4 smartphones. We performed the same experiment as described above using DP images captured from a Pixel 4 phone. The result is shown in the last

column of Figure B.3. Due to the small lens and sensor size, smartphone images have notably more noise than their DSLR counterparts. Radial distortion is also more pronounced. As a consequence, the kernel estimates are more noisy compared to the Canon DSLR. However, it can be observed that the symmetry of the kernels still holds to a fair extent.

B.4 Proposed Method

In this section, we first introduce our optimization-based algorithm for depth estimation from DP data. The loss to be minimized is unsupervised, and is constructed based on the kernel symmetry property described in Section B.3. We also show that substituting our optimization method with a CNN offers considerable speed-up.

B.4.1 Unsupervised Loss Based on Dual-Pixel Kernel Symmetry

Using equations (B.1) and (B.2), and the associative and commutative properties of convolution, it can be shown that $\mathbf{G}_l * \mathbf{H}_r = \mathbf{G}_r * \mathbf{H}_l$ as follows:

$$\begin{aligned} \mathbf{G}_l * \mathbf{H}_r &= (\mathbf{F} * \mathbf{H}_l) * \mathbf{H}_r && \text{from equation B.1} \\ &= \mathbf{F} * (\mathbf{H}_l * \mathbf{H}_r) && \text{associative property} \\ &= \mathbf{F} * (\mathbf{H}_r * \mathbf{H}_l) && \text{commutative property} \\ &= (\mathbf{F} * \mathbf{H}_r) * \mathbf{H}_l && \text{associative property} \\ &= \mathbf{G}_r * \mathbf{H}_l && \text{from equation B.2} \end{aligned} \tag{B.5}$$

This relationship [213] has been used for classic depth from defocus [80, 214]. Here, we apply it to the case of DP defocus kernels. Using the kernel symmetry property ($\mathbf{H}_l = \mathbf{H}_r^f$ defined earlier in Section B.3), this can be rewritten as $\mathbf{G}_l * \mathbf{H}_r = \mathbf{G}_r * \mathbf{H}_r^f$. Formally, our unsupervised loss function can now be expressed as

$$\begin{aligned} &\arg\min_{\mathbf{H}_r} \{\mathbf{G}_l * \mathbf{H}_r - \mathbf{G}_r * \mathbf{H}_r^f\}, \\ &\text{subject to } \mathbf{H}_r \geq \mathbf{0}, \sum \mathbf{H}_r = \frac{1}{2}, \end{aligned} \tag{B.6}$$

where non-negativity and sum to half constraints are imposed since $\mathbf{H}_r$ is a blur kernel corresponding to one half of the aperture. The loss function states that the left DP patch

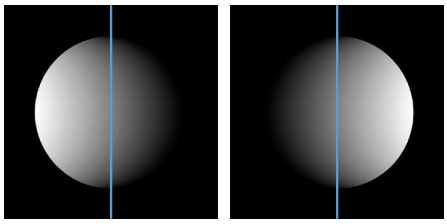


Figure B.4: Our parameterized translating disk kernels corresponding to the left and right DP sub-aperture views.

blurred with the right DP kernel equals the right DP patch blurred with the *flipped* right DP kernel. To further simplify the optimization, we parameterize the blur kernel as a translating disk (see Figure B.4) such that the (signed) radius of the disk s is the only free parameter to be estimated. Formally, we express the parameterized blur kernel $\mathbf{H}_{r_s}$ as

$$\mathbf{H}_{r_s} = \sum_{i=0}^{|2s|} \mathbf{C}\left(\frac{s}{|s|}i, 0\right), \tag{B.7}$$

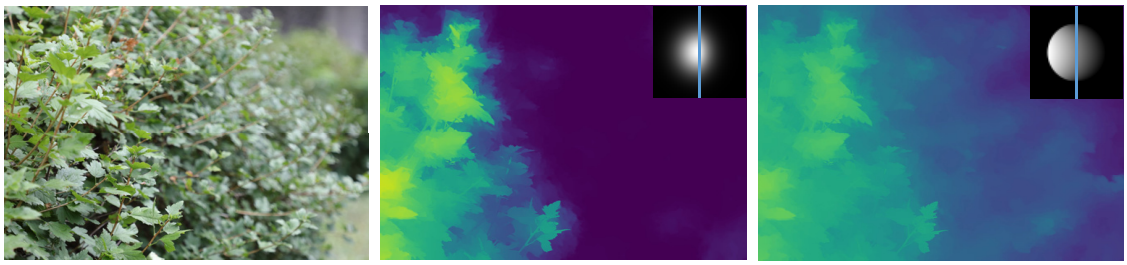
where $|.|$ denotes absolute value, and $\mathbf{C}(x, y)$ represents a circular disk of radius $|s|$ centered at pixel coordinates (x, y). We normalize $\mathbf{H}_{r_s}$ to sum to half. Equation (B.6) can now be rewritten as

$$\arg\min_{s} \{\mathbf{G}_l * \mathbf{H}_{r_s} - \mathbf{G}_r * \mathbf{H}^f_{r_s}\}. \tag{B.8}$$

Note that the disparity is directly proportional to the signed blur kernel radius. The sign disambiguates whether the depth region corresponding to the patches $\mathbf{G}_l$ and $\mathbf{G}_r$ is in front of or behind the focal plane. As already discussed in Section B.1, from DP data, it is only possible to recover this disparity which is related to the actual depth by an affine transformation [25].

Our unsupervised loss of equation (B.8) holds only for a constant-depth region of the scene. Therefore, given a test image, we adopt a sliding-window approach similar to [27] to estimate the depth map. The details of these steps are provided in Section B.4.3.

In the literature, defocus PSFs have most commonly been parameterized using a 2D Gaussian function. We also experimented with a *decaying* version of the Gaussian kernel (see Figure B.5) to parameterize $\mathbf{H}_{r_s}$. However, we found from our experiments that our proposed translating disk kernel yields more accurate depth estimates. An example is



(a) Input image (b) Depth map using a decaying Gaussian kernel (c) Depth map using the proposed kernel

Figure B.5: A comparison between a decaying Gaussian kernel and our proposed translating disk kernel. Our translating disk kernel more closely models the PSF observed on real DP data, and produces a more accurate depth map. Note that only the image is shown; the left and right DP views that are input to equation (B.8) have not been presented. Note that the depth map is slightly blurry due to our new PSF and the post-processing edge-aware filtering applied.

shown in Figure B.5. Note that the depth map is slightly blurry due to our new PSF and the post-processing edge-aware filtering applied.

B.4.2 CNN for Faster Inference

One limitation of our approach of Section B.4.1 is the need to solve an optimization problem at each pixel location. This makes inference slow when performed over the entire image. To improve the runtime performance, we can substitute our optimization algorithm with a CNN.

To generate training data for our CNN, we imaged 12 printed postcards at different focus settings using the Canon EOS 5D Mark IV DSLR camera. Of these, 10 postcards were used for training, while the remaining two were used for validation. The postcards were placed at a fixed distance fronto-parallel to the camera, and the focus setting was varied to introduce different levels of defocus blur. We chose 10 focus settings to image each postcard, giving us a total of 120 images. The focus settings themselves were selected such that an approximately equal number had the focal plane behind and in front of the postcard plane. The left and right DP views were extracted from all 120 images. Patches lying in the central 66% region were cropped from both views (to avoid radial distortion

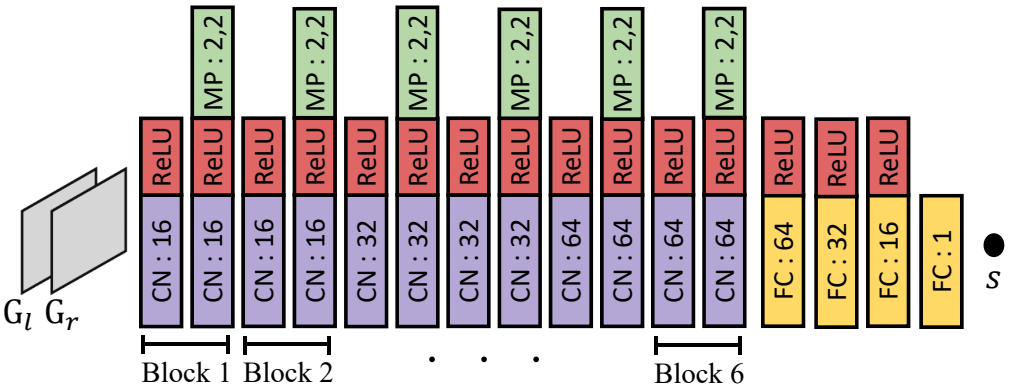


Figure B.6: Our network architecture. The network takes a six-channel input containing the left ($\mathbf{G}_l$) and right ($\mathbf{G}_r$) DP patches (each comprising three color channels) stacked together. CN : z denotes a 2D convolution layer with z filters of size 3×3 and a stride of one. ReLU represents a rectified linear unit activation layer. MP : p, q denotes a 2D max-pool layer with a kernel size p and a stride q. There are 6 blocks of CN, ReLU, and MP. FC : v is a fully connected layer of length v. The network outputs the signed radius value s of the disk kernel.

effects) to generate training and validation data. We used patches of size 111×111 pixels. After filtering out homogeneous patches, we generated in total $\sim 17,500$ patches for training and another $\sim 2,100$ patches for validation.

Our network architecture is shown in Figure B.6. The network takes the left and right DP patches, $\mathbf{G}_l$ and $\mathbf{G}_r$, respectively, as input, and outputs the signed radius s of the disk kernel. The ground truth labels (i.e., the value of s) needed for training the network are generated by running our optimization algorithm (equation (B.8) of Section B.4.1) on the 120 images. Note that the labels need to be generated only per image and not per patch. This is because all patches in a given image are at a constant depth from the camera and experience the same amount of defocus blur since the postcards are arranged fronto-parallel to the camera.

Similar to the optimization method of Section B.4.1, we adopt a sliding-window approach at test time to recover the depth map of the scene. For comparison, while our optimization approach takes approximately 20 minutes to estimate the disparity values, our CNN takes under 10 seconds for disparity prediction on a 3-megapixel image. Fol-

lowing [27], we apply bilateral space techniques as a final processing step to ensure that the depth map is edge-aligned with the corresponding input image. We provide details of these steps in the next section.

B.4.3 Edge-Aware Filtering

Our kernel symmetry assumption holds only for a constant-depth region of the scene. To obtain the depth map given a test image, we apply a sliding-window approach similar to [27]. The patch size, for all experiments in this appendix for both our optimization-based approach (Section B.4.1) and our CNN-based approach (Section B.4.2), is fixed to 111×111 pixels, and the stride is set to 33. Our disparity estimates can be noisy, particularly in homogeneous regions, or near depth boundaries. As a result, we compute a confidence value M for each window as:

$$M = S \times e^{-\beta E}, \tag{B.9}$$

where S is the average of the horizontal Sobel operator computed over the left and right DP patches, and E is the estimation error $\mathbf{G}_l * \mathbf{H}_{r_s} - \mathbf{G}_r * \mathbf{H}^f_{r_s}$. We use the horizontal Sobel operator because the disparity is along the horizontal axis, and only vertical edges provide meaningful information. While the Sobel value S weights down the confidence in homogeneous regions, the term $e^{-\beta E}$ decreases the confidence around depth boundaries where the error E is expected to be high. Following [27], we use the bilateral solver of [215] to make our estimated depth map smooth and edge-aware. Given the confidence map, the input image, and our estimated depth map, the bilateral solver outputs an edge-aware depth map. We perform guided filtering [216] of this depth map to produce our final output. We follow these same edge-aware filtering steps for both our optimization-based algorithm and our CNN.

B.5 Experiments

To the best of our knowledge, there are no publicly available datasets that provide dual-pixel data with corresponding ground truth depth maps. Therefore, we captured a dataset using a dual-pixel camera to evaluate our algorithm's performance. We perform both quantitative and qualitative evaluation. We use the Canon camera for quantitative analysis since the DSLR provides greater flexibility and control in capturing focal stacks, which we

use to obtain the ground truth depth maps. Qualitative comparisons are performed using data from both the Canon DSLR and the Pixel 4 smartphone.

The remainder of this section is organized as follows. We first provide details on CNN training. Next, we discuss comparison methods and the metrics used for evaluation. We then describe how we capture focal stacks and generate the ground truth depth maps for quantitative evaluation. Lastly, qualitative results on DSLR as well as smartphone DP data are presented.

B.5.1 CNN Training and Implementation Details

We adopt He's weight initialization [100], and use the Adam optimizer [90] to train our model. The initial learning rate is set to $1{\times}10^{-5}$, which is decreased by half every 20 epochs. We train our model with minibatches of size 10 using the mean squared error (MSE) loss. Our network is trained using Keras [101] with TensorFlow [102] on an NVIDIA TITAN X GPU. Our model has approximately 176K parameters and is trained for 100 epochs.

We used a fixed value of $\beta = 10^{-6}$ in equation (B.9). The codes for the bilateral solver [215] and the guided filter [216] have been made publicly available by the authors. For the bilateral solver, the hyper-parameters were chosen as $\sigma_{xy} = 16, \sigma_l = 16, \sigma_{uv} = 8, \lambda = 128$ and 25 iterations of preconditioned conjugate gradient (PCG) for all experiments. We used $r = 10$ and $\epsilon = 10^{-6}$ for the guided filter.

B.5.2 Comparison With Other Methods

The work most closely related to ours is the DP depth estimation method of [25]. At the time of submitting this work, neither the dataset nor the trained model/code of [25] is publicly available. As such, we compare our results against the state-of-the-art deep learning-based stereoscopic and monocular depth estimation techniques of [108, 193, 197]. The codes for these three methods have been made available by the authors. Of these, the networks of [108, 197] are trained through self-supervision using stereo pairs, while the method of [193] uses ground truth depth for supervised training. The authors of [25] retrained methods such as [108] that are based on stereo self-supervision on the two DP views from their dataset. However, the dataset of [25] is not publicly available. Moreover, our method, by virtue of being unsupervised, does not require the collection of a large dataset. Therefore, we report results using the best performing models for [108, 197]

without any retraining. We also compare against the method of Wadhwa *et al.* [27] that retools classic stereo techniques to recover depth from dual pixels. Since their code is not available, we implement their method based on their description. We use the same parameters recommended by the authors for our experiments on the Pixel smartphone. For DSLR images, the disparity will be higher, and so we increase the tile size as well as the sum of squared differences (SSD) search window (see Section 4.1 of [27]) for optimal performance.

B.5.3 Error Metrics

As mentioned in Section B.1, dual-pixel data has a fundamental ambiguity in that depth be recovered only up to an unknown affine transformation [25]. Thus metrics such as mean absolute error (MAE) or root mean squared error (RMSE) that measure error between ground truth and estimated depth in absolute terms cannot be applied. Following [25], we use affine invariant versions of MAE and RMSE, denoted AI(1) and AI(2), respectively. We can define AI(p) as

$$\arg\min_{a,b} \left(\frac{\sum_{(x,y)} |D^*(x,y) - (a\hat{D}(x,y) + b)|^p}{N} \right)^{\frac{1}{p}}, \tag{B.10}$$

where N is the total number of pixels, $\hat{D}$ is the estimated disparity, and D^* is the ground truth inverse depth. Note that [25] used a weighted variant AIWE(p) of the above equation, where the weights or confidence values are computed based on the coherence across views in their stereo rig. Since we compute our ground truth depth maps from focal stacks using DFD, we do not include this weighting term in our calculation. AI(2) can be formulated as a straightforward least-squares problem, while AI(1) can be computed using iterative reweighted least squares (we used five iterations in our experiments). We also use Spearman's rank correlation ρ_s, which evaluates ordinal correctness between the estimated depth and the ground truth, for evaluation. See [25] for more details. Once again, we differ from [25] in that they use a weighted variant of Spearman's ρ.

B.5.4 Quantitative Evaluation

To quantitatively evaluate our method's performance, we need DP data paired with ground truth depth information. We compute "ground truth" depth maps by applying well-established depth-from-defocus techniques on focal stacks captured using a DP camera.

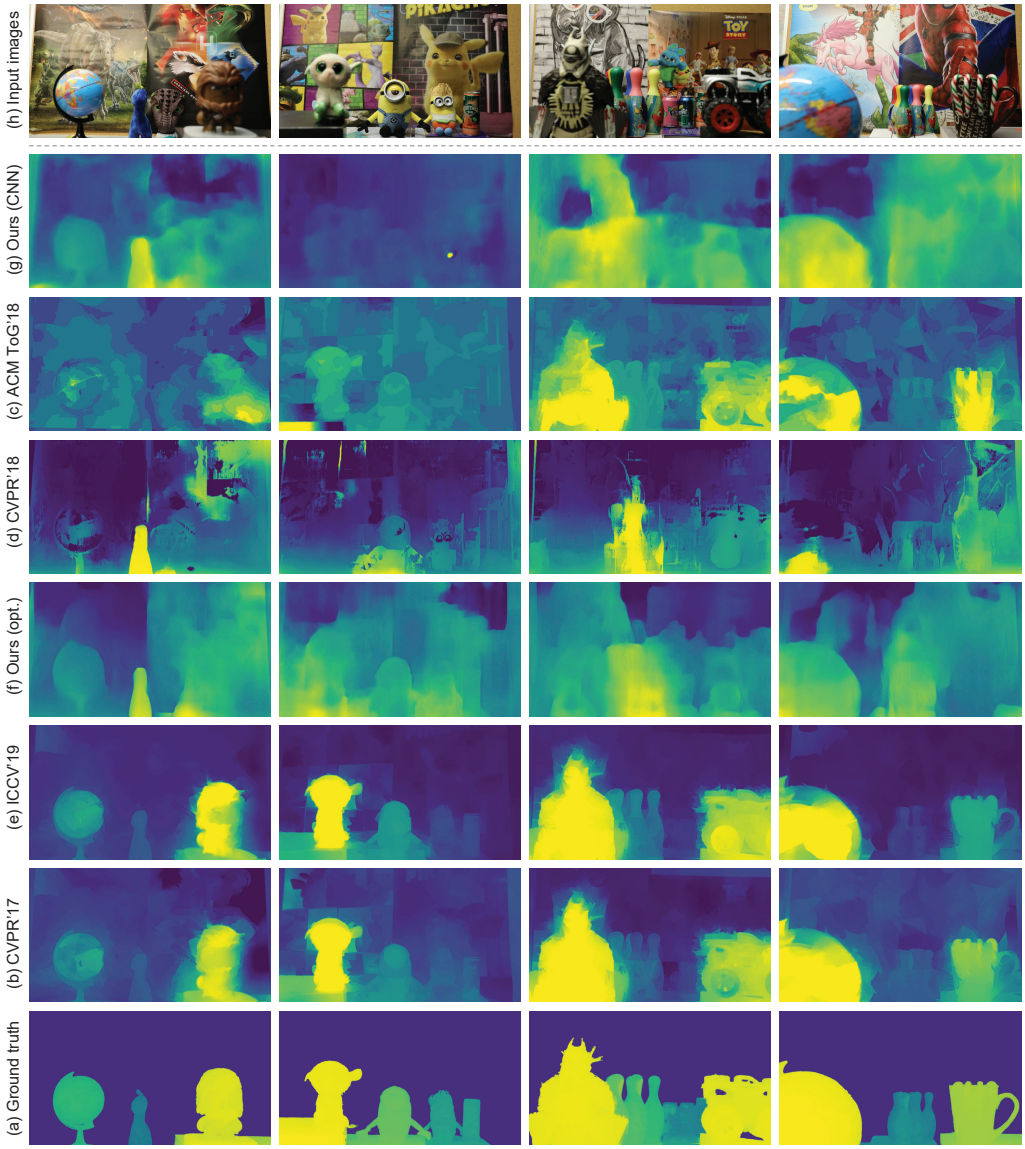


Figure B.7: Input images (a) from our focal stack dataset, the results of the deep learning-based methods of [108] (b), [193] (d), [197] (e), the output of the traditional stereo algorithm of [27] (c), the results of our optimization-based approach (f) and our CNN-based approach (g), and the ground truth (h). An affine transform has been applied to all visualizations to best fit the ground truth. The deep learning-based algorithms [108, 193, 197] do not perform well in general. The stereo approach of [27] fares better in comparison but has many errors in the background. In contrast, both our approaches are able to separate out the foreground objects from the background more accurately. Best seen zoomed-in in an electronic version.

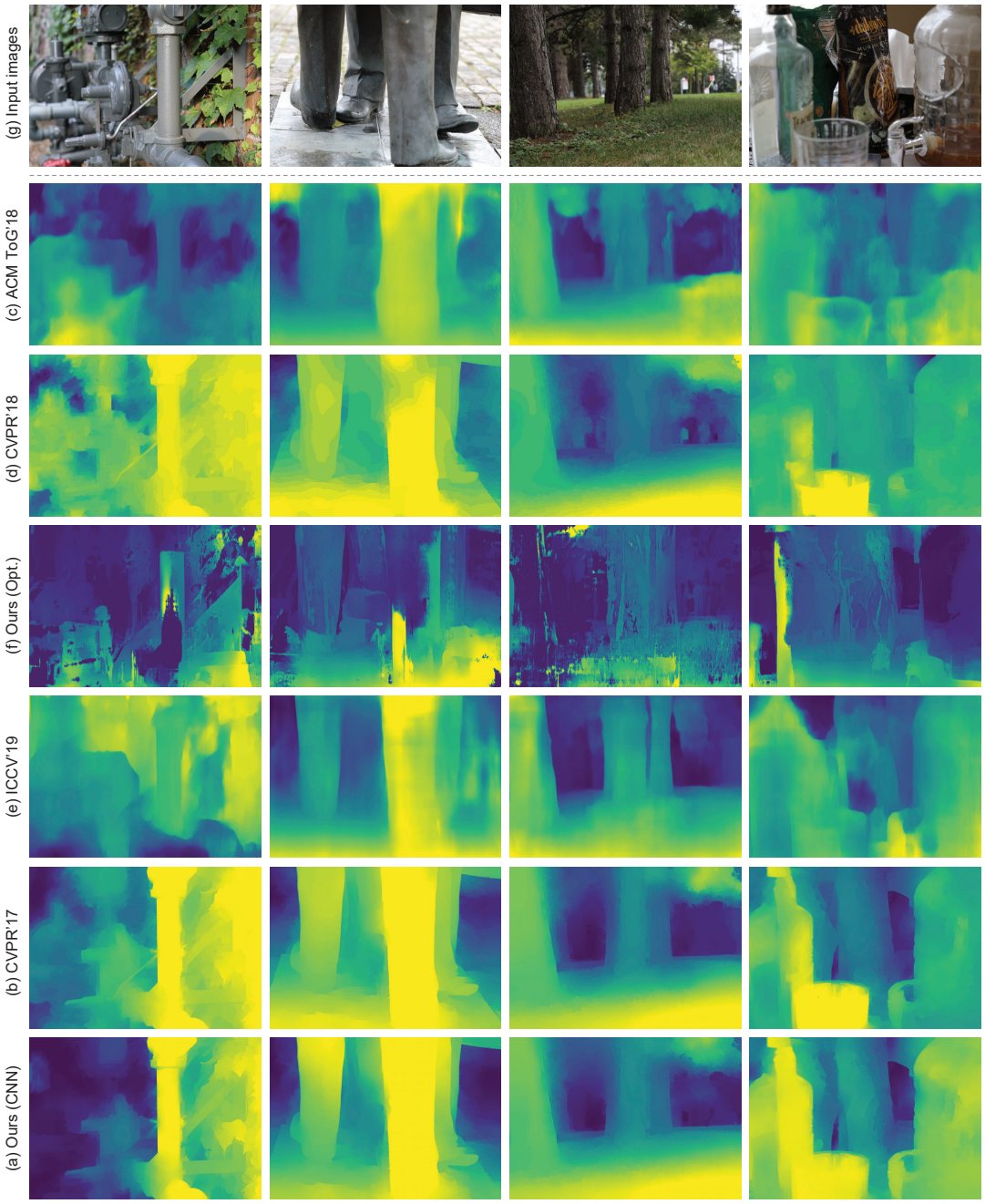


Figure B.8: Qualitative results of our proposed method as well as competing algorithms on data captured using the Canon EOS 5D Mark IV DSLR camera.

Table B.1: Accuracy of different methods on our dataset. Lower is better. Best results are shown in bold. The right-most column shows the geometric mean of all the metrics.

Method	AI(1)	AI(2)	$1 - \lvert\rho_s\rvert$	Geometric Mean
CVPR'17 [108]	0.1175	0.1865	0.7454	0.2488
ACM ToG'18 [27]	0.0875	0.1294	0.2910	0.1443
CVPR'18 [193]	0.1082	0.1788	0.6235	0.2250
ICCV'19 [197]	0.1139	0.1788	0.6153	0.2285
Ours (optimization)	**0.0469**	**0.0742**	0.0817	0.0646
Ours (CNN)	0.0481	0.0743	**0.0779**	**0.0645**

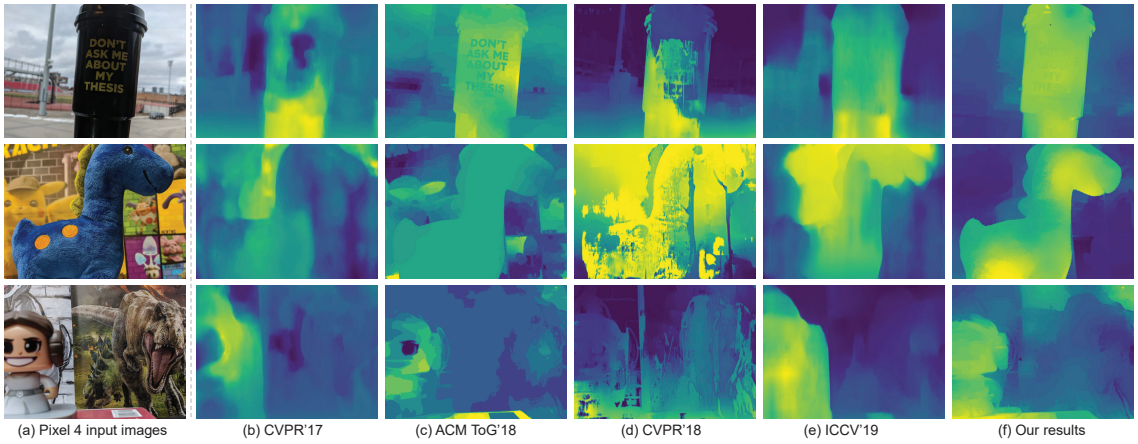


Figure B.9: Qualitative results of our proposed method as well as competing algorithms on data captured using the Pixel 4 smartphone.

Alternate approaches to obtaining ground truth depth include using direct depth sensors, such as Kinect or LIDAR, or, as in [25], building a custom camera rig and applying multi-view stereo methods. However, both these approaches involve cumbersome registration and synchronization procedures – in the former case, between the DP camera and the direct depth sensors, and in the latter, between the central DP camera and the other stereo cameras in the rig. In contrast, DFD techniques can be applied directly to focal stacks captured using a single camera, and do not require additional cameras or sensors. One limitation of using DFD is that the scene being imaged has to remain unchanged for the entire duration of the capture of the stack. This precludes scenes or objects that are dynamic. While our quantitative evaluations are thereby limited to static scenes, we would like to note that we perform qualitative studies on images captured under unconstrained settings.

We captured 10 focal stacks corresponding to different scenes. We used the Canon EOS 5D Mark IV DSLR camera, which allows fine-grained control of the focus settings, to capture the stacks. Each stack contains between 75 and 90 images. Objects in the scenes were placed between 0.5 m to 2 m to ensure there is interesting nearby depth variation. To make the dataset challenging, we used printed posters (different from the postcards used in Section B.4.2 to generate training data for the CNN) with fairly diverse and complex textures as background, while the foreground objects were designed to have occlusions and clutter. Sample images are shown in the first row of Figure B.7. To estimate the ground truth depth maps from the focal stacks, following [217], we used the commercially available and widely used HeliconSoft software. We noticed that the depth map produced by HeliconSoft tends to have errors in large homogeneous regions. This is a limitation of DFD methods in general since focus operators rely primarily on texture. Therefore, we manually annotated the depth values at these few pixels. The ground truth depth maps obtained from the focal stacks are shown in the last row of Figure B.7.

For quantitative evaluation, we selected 10 images at random from each focal stack for a total of 100 images. Although all images in the stack contain DP information, we extract the left and right DP views only from these selected images. For the purpose of estimating depth maps using DFD, we treat all images as conventional RGB images. The results of our method, as well as comparisons, on a few representative examples from our focal stack dataset are shown in Figure B.7. The deep learning methods of [108,193,197] are given the image as input, while [27] and our method take the left and right DP views as input. The

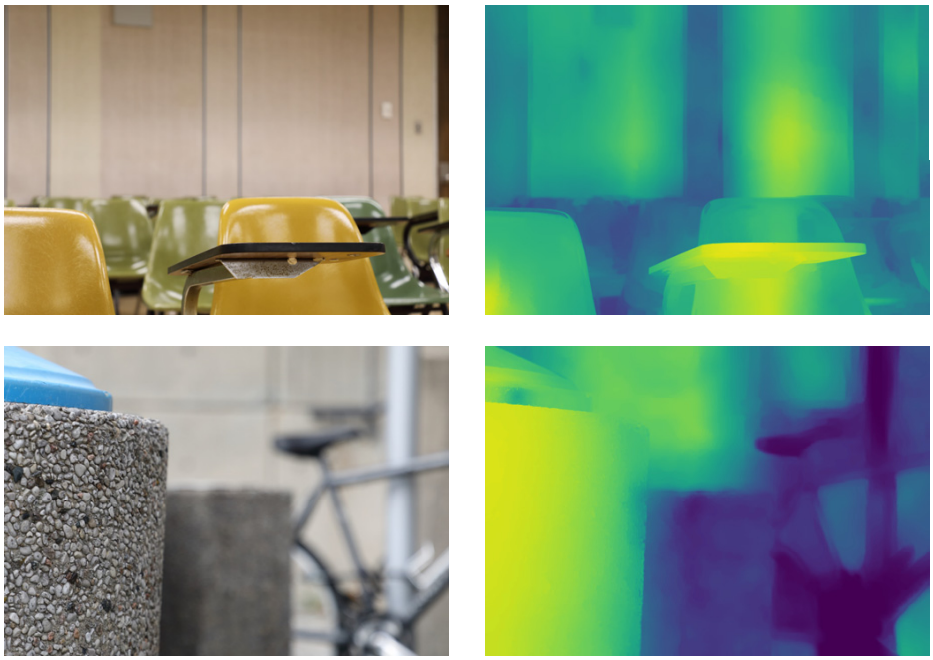


Figure B.10: Example failure cases of our algorithm. Homogeneous depth planes such as the plain background walls in the two images can lead to erroneous depth estimates.

methods of [108, 193, 197] perform poorly and fail to distinguish the foreground objects in most cases. The unsupervised stereo technique of [27] also produces erroneous depth estimates particularly in the first two examples. In comparison, our optimization-based and CNN-based approaches generate more accurate depth maps. Quantitative results averaged over the 100 images in our dataset are reported in Table B.1. It can be observed that our proposed method outperforms all competing algorithms on all three metrics. While our optimization approach yields better performance in terms of AI(1) and AI(2), our CNN records a slightly lower (better) Spearman's rank correlation.

B.5.5 Qualitative Evaluation

Qualitative results of our proposed method as well as competing algorithms on data from the Canon DSLR camera and the Pixel 4 smartphone are provided in Figs. B.8 and B.9, respectively. It can be seen that our proposed method produces more accurate depth

estimates than competing methods. While our focal stack dataset was collected indoors in a controlled environment, the examples in Figure B.8 also include outdoor scenes captured in unconstrained settings. In the Pixel 4, the DP data is embedded in the green channel [25] – that is, the DP views are single-channel as against three color channels in the case of the Canon DSLR. We do not run our CNN method on Pixel data because our network was trained on Canon images with a six-channel input. In general, we found that Pixel data is more challenging, particularly for far-away scenes. This is mainly because the data has higher levels of noise compared to the DSLR (as observed in Figure B.3), and the disparity is lower due to the small aperture size. However, our method still produces fairly good depth estimates as seen from our results in Figure B.9.

Failure cases As with almost all algorithms that use focus/defocus cues for depth inference, our method too relies on the presence of texture. Large homogeneous regions and textureless surfaces that do not contain useful information for defocus estimation can lead to errors. Two failure cases of our method are shown in Figure B.10.

B.6 Conclusion

This appendix has examined the image formation in dual-pixel sensors and proposed a parametric PSF to model the defocus-disparity in the two sub-aperture views. Currently, this DP sensor data can be obtained only on a limited number of devices – namely the Canon EOS 5D Mark IV and the Pixel 3 and 4 cameras. We have shown our model is applicable to both DLSRs with a large aperture and smartphone devices with a small and fixed aperture. Using our parametric model, we described how to leverage the symmetry property between the two corresponding PSFs to formulate an unsupervised loss. We demonstrated the efficacy of this loss in an optimization framework for the task of depth estimation from DP data. Experiments show the effectiveness of our PSF formulation of disparity. While a CNN was introduced to speed up our inference, future work aims to use the unsupervised loss to directly train a CNN in an end-to-end manner.

Appendix C

PSF Calibration and Estimation

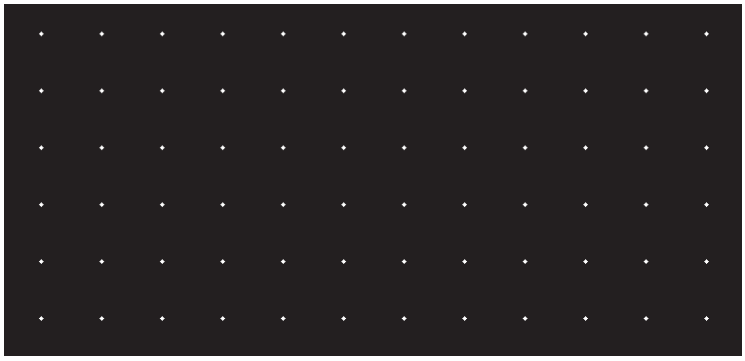
In this appendix, we introduce the calibration procedure used to estimate the point spread functions (PSFs) (Section C.1) along with the parameter range searching for our parametric dual-pixel (DP) PSF model (Section C.2). Next, another calibration procedure is described for estimating the radial distortion coefficients (Section C.3).

C.1 PSFs on a Real Dual-Pixel Sensor

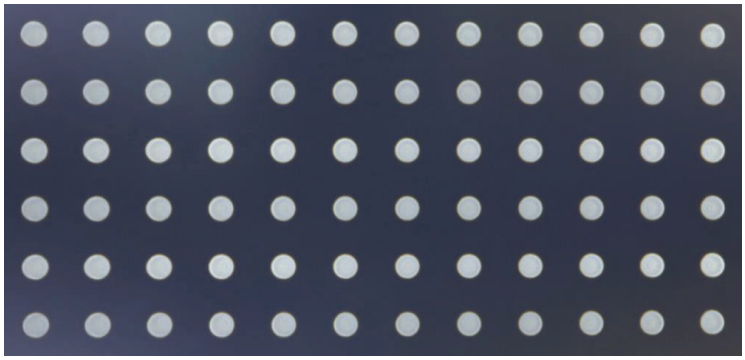
As mentioned in Section 6.2.2, we aim to estimate more accurate and realistic DP-PSFs; thus, we follow the same calibration practice as in [4, 211, 218]. The calibration pattern contains a grid of small disks with known radius and spacing as shown in Figure C.1-A. The pattern in Figure C.1-A is computer-generated, and we display it on a 27-inch LED display of resolution 1920×1080. Then, we use the Canon 5D Mark IV DSLR camera for our capturing procedure, as it facilitates reading out DP data. The LED display is placed parallel to the image plane and at a fixed distance of about one meter.

We captured many images by varying the camera parameters – namely, focus distance, aperture size, and focal length – covering a wide range of shape varying PSFs (an example image is shown in Figure C.1-B). Since we apply radial distortion to our synthetically generated data, we seek to estimate the least radially-distorted PSF, that in practice, is found close to the image center. Patches fully containing the disks are identified, and the center of the disks estimated by finding the centroid of these patches. The radius of computer-generated disks is a known fraction of the distance between disk centers.

Similar to [4,211], we adopt a non-blind PSF estimation, in which the latent sharp disk



(A) Computer-generated calibration pattern. A grid of disks of known radius



(B) Out-of-focus calibration pattern as imaged by Canon 5D Mark IV DSLR camera

Figure C.1: Calibration patterns are used for estimating dual-pixel (DP) point spread functions (PSFs). A: our synthesized computer-generated pattern of a grid of small equal-size disks. B: the same calibration pattern as imaged by Canon 5D Mark IV DSLR camera. Note that the pattern captured in (B) is out-of-focus.

$\mathbf{S}$ is known. Then, the PSF from the camera $\mathbf{E}$ is estimated using $\mathbf{S}$ and the corresponding blurred patch $\mathbf{B}$ by solving:

$$\begin{aligned} \arg\min_{\mathbf{E}} \textstyle\sum_i \left\| \mathbf{D}_i(\mathbf{S} * \mathbf{E} - \mathbf{B}) \right\|_2^2 + \left\| \mathbf{E} \right\|_1, \\ \text{subject to } \mathbf{E} \geq \mathbf{0}, \end{aligned} \tag{C.1}$$

where $\mathbf{D}_i \in \{\mathbf{D}_x, \mathbf{D}_y, \mathbf{D}_{xx}, \mathbf{D}_{yy}, \mathbf{D}_{xy}\}$ denotes the spatial vertical and horizontal derivatives. The ℓ_1-norm of $\mathbf{E}$ encourages sparsity of the PSF entries. Additionally, another non-negativity constraint is imposed on the entries of $\mathbf{E}$. A wide range of PSFs for each DP view is estimated independently. Figure C.2 shows examples of the estimated DP-

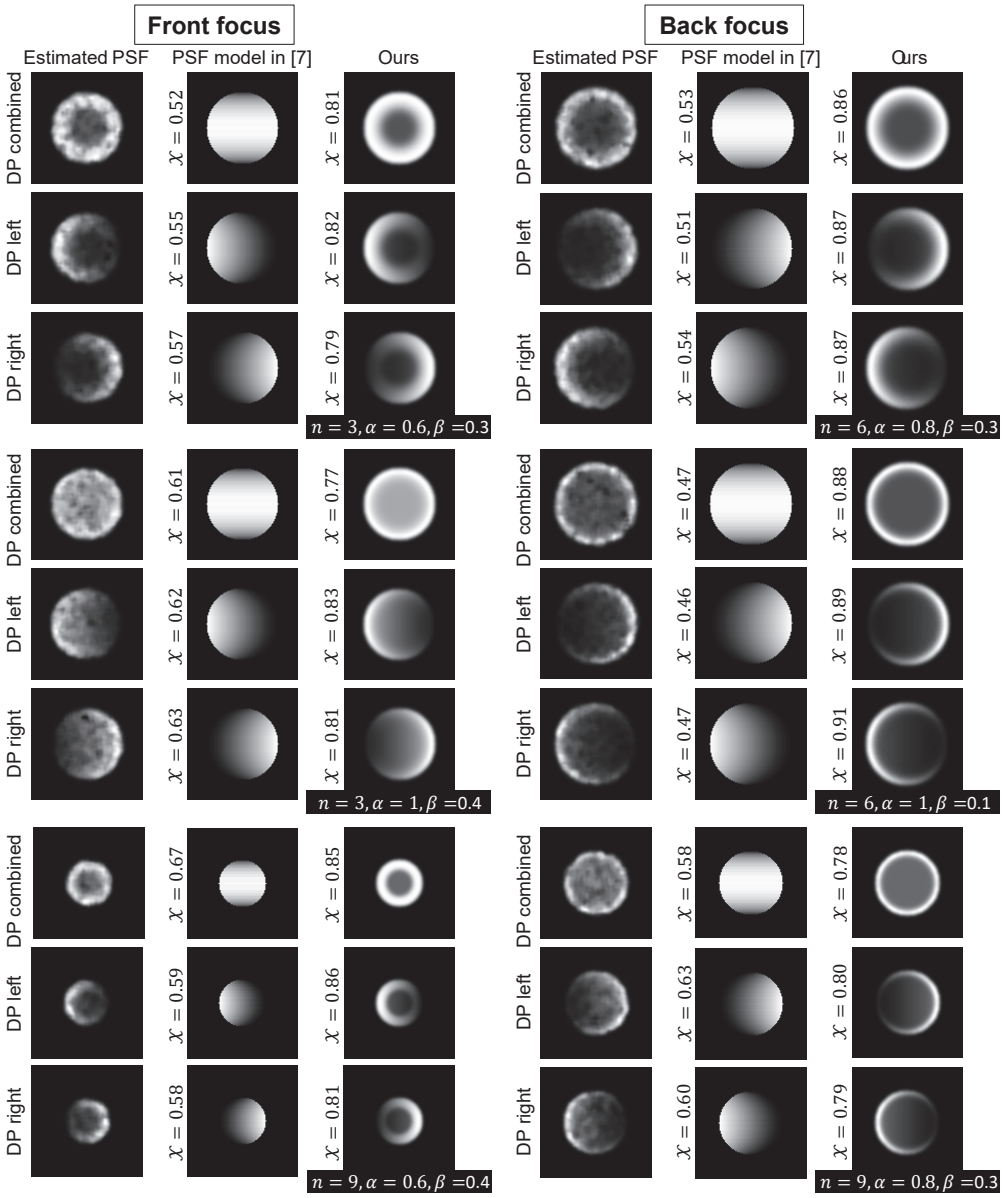


Figure C.2: The estimated point spread functions (PSFs) in comparison with the dual-pixel (DP) PSF model in [4] and our parametric DP-PSF model. The PSFs for the DP combined, left, and right are estimated independently. The similarity with the estimated PSF is measured using the 2D cross-correlation $\mathcal{X}(.)$ and is shown on the left for each case. The parameters (i.e., n, α, β) used to generate our DP-PSF are shown below each case. Our parametric DP-PSF model obtains higher similarity with the estimated ones.

PSFs.

C.2 Parameter Search for Our Dual-Pixel PSFs

In Section 6.2.2 and Section 6.5, we also mentioned a mechanism to select the effective range of parameters (i.e., n, α, β, κ) for our parametric DP-PSF modeling. Recall that n is the Butterworth filter order, and α is used to control its 3dB cutoff position. β is the filter lower bound scale, and κ is the Gaussian smoothing factor.

Our goal is not to exactly match the measured PSFs found in cameras but rather to find representative PSFs that are very similar to what is estimated. The main reason is that we used a single camera, and as shown in Figure C.2, the estimated PSFs are noisy and not perfectly circular. This observation is expected due to camera-specific physical constraints like the positioning of the microlens, depth of the sensor wells, and other optics manufacturing imperfections. Therefore, we limit the parameter search to a range of discrete values for each parameter sampled as:

$$\begin{aligned} n &\in \{1, 2, 3, \ldots, 15\}, , \\ \alpha, \beta &\in \{0.1, 0.2, \ldots, 1.0\}, \\ \kappa &\in \{0.14, 0.21, \ldots, 0.42\}. \end{aligned} \tag{C.2}$$

Then, we perform a brute-force search in this bounded space by solving the following equation:

$$\underset{n,\alpha,\beta,\kappa}{\arg\min} \sum_i \left\| \mathbf{D}_i(\mathbf{E} - \mathbf{H}(n, \alpha, \beta, \kappa)) \right\|_2^2, \tag{C.3}$$

where $\mathbf{E}$ is the estimated PSF from a real camera and $\mathbf{H}(n, \alpha, \beta, \kappa)$ is our parameterized PSF. We found the parameters achieve the highest similarity at:

$$\begin{aligned} n &\in \{3, 6, 9\}, \\ \alpha &\in \{0.4, 0.6, 0.8, 1.0\}, \\ \beta &\in \{0.1, 0.2, 0.3, 0.4\}, \\ \kappa &= 0.14. \end{aligned} \tag{C.4}$$

Given the best values we defined for each parameter, we can generate 48 combinations of possible PSFs, and those represent our bank of DP-PSFs used to generate our synthetic

DP data. Figure C.2 shows examples of our parameterized PSFs in comparison with the estimated ones. Our PSFs demonstrate much higher correlation with the estimated real PSFs compared to the DP-PSF model in [4]. The similarity is measured using the 2D cross-correlation $\mathcal{X}(.)$.

Such comprehensive PSF calibration (Section C.1) and parameter search (Section C.2) is not possible for smartphone cameras due to uncontrollable camera factors like aperture size and focal length. Additionally, up to our knowledge, only Pixel 3 and 4 smartphones allow direct access to the DP data, and the Pixel-DP API provided in [25] does not facilitate manual focus, which limits us from controlling the focus distance as well. The work in [4] estimated two PSFs from a Pixel 3 smartphone, and we found that they achieve a high correlation with our parametric PSF model.

C.3 Radial Distortion Coefficients

Based on Section 6.2.3, radial distortion is applied for more realistic imagery since the input images and our parametric DP-PSFs are not radially distorted. To this aim, we use the division model [115], as follows:

$$(x_d, y_d) = (x_o, y_o) + \frac{(x_u - x_o, y_u - y_o)}{1 + c_1 R^2 + c_2 R^4 + \cdots}, \tag{C.5}$$

where (x_u, y_u) and (x_d, y_d) are the undistorted and distorted pixel coordinates respectively, and c_i is the i^{th} radial distortion coefficient. R is the radial distance from the image plane center (x_o, y_o). This section introduces the calibration used to capture real-world radial distortion cases and is followed by the coefficient search procedure used to mimic real-world radial distortion. Recall that radial distortion is associated with zoom lenses and depends mainly on the camera's focal length.

We synthesize a uniform pattern of squares, as shown in Figure C.3-A. We follow the same setup described in Section C.1. Still, with the following changes: (1) we capture an in-focus calibration pattern, (2) the focal length is changed across captures and the aperture remains fixed, (3) the distance between the display and camera is adjusted accordingly to make sure the full-resolution image is within camera's field of view, sine increasing the focal length introduces zoom/magnification effect, and (4) the focus distance is also adjusted accordingly to make sure the pattern is in focus. Figure C.3-B shows examples of the calibration pattern as imaged by the Canon camera at different focal lengths.

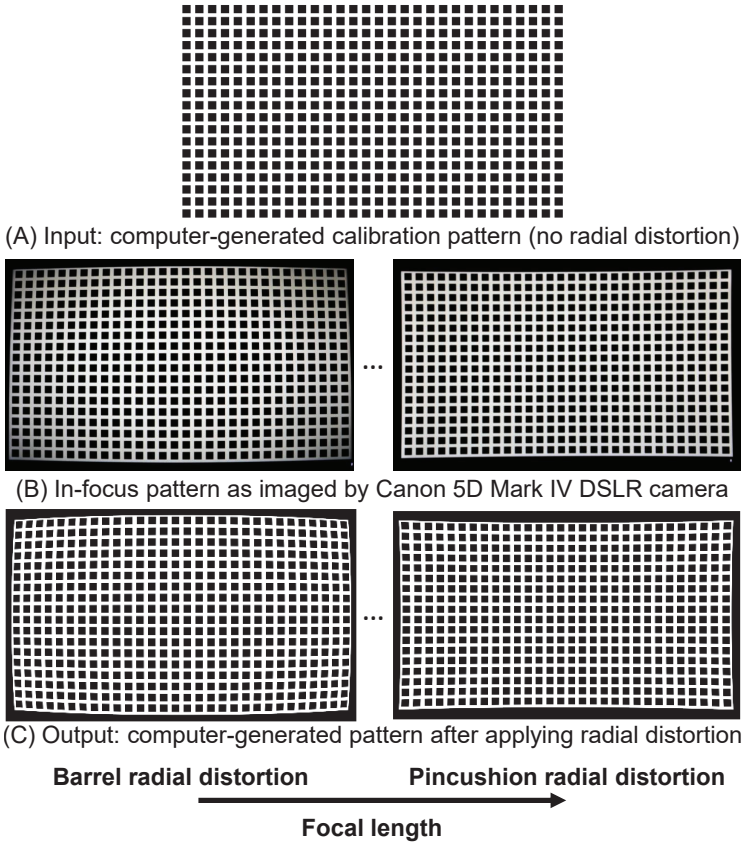


Figure C.3: Calibration pattern used for estimating the radial distortion coefficients. A: the input computer-generated pattern with no radial distortion applied. B: the same pattern as imaged by the Canon 5D Mark IV camera at different focal lengths. C: the computer-generated pattern after applying radial distortion.

We performed five captures at five different focal lengths ranging from min to max. Each is mapped to a focal length in our predefined five camera parameter sets – namely: $\{4, 5, 6\}, \{5, 8, 6\}, \{7, 5, 8\}, \{10, 13, 12\}, \{22, 10, 30\}$ — such that each set represents focal length, aperture size, and focus distance. With these five representative radial distortions that cover barrel as well as pincushion distortions, a brute-force search is performed to find the c_i coefficients that satisfy the following:

$$\underset{c_1,c_2,c_3}{\arg\min}\, \mathcal{X}(\mathbf{I}_f, \mathbf{I}_d(c_1, c_2, c_3)), \tag{C.6}$$

where $\mathbf{I}_f$ is the calibration pattern as imaged by the Canon camera at a certain focal length f (e.g., Figure C.3-B). $\mathbf{I}_d(c_1, c_2, c_3)$ is the computer-generated input pattern but after

applying radial distortion based on the coefficients c_1, c_2, c_3 (see example in Figure C.3-C). Since we have few examples, $\mathbf{I}_f$ is re-centered manually to match $\mathbf{I}_d(c_1, c_2, c_3)$'s center. While Equation C.5 can be defined with more coefficients, we found three coefficients sufficient to approximate our real-world distortion examples. The final optimal five sets of coefficients are:

$$\begin{gathered}
\{2 \times 10^{-2}, 2 \times 10^{-2}, 3 \times 10^{-2}\}, \\
\{8 \times 10^{-3}, 2 \times 10^{-3}, 2.2 \times 10^{-3}\}, \\
\{-4 \times 10^{-3}, 9 \times 10^{-4}, -9 \times 10^{-4}\}, \\
\{-7 \times 10^{-3}, -3.8 \times 10^{-3}, -3.6 \times 10^{-3}\}, \\
\{-8 \times 10^{-3}, -5 \times 10^{-3}, -4.5 \times 10^{-3}\}.
\end{gathered} \tag{C.7}$$

Appendix D

CIE XYZ: Unprocessing Images for Defocus Estimation

Cameras currently allow access to two image states: (i) a minimally processed linear raw-RGB image state (i.e., raw sensor data) or (ii) a highly-processed nonlinear image state (e.g., sRGB). There are many computer vision tasks that work best with a linear image state, such as image deblurring and image dehazing. Unfortunately, the vast majority of images are saved in the nonlinear image state (e.g., our DP canon dataset in Chapter 5). Because of this, a number of methods have been proposed to "unprocess" nonlinear images back to a raw-RGB state. However, existing unprocessing methods have a drawback because raw-RGB images are sensor-specific. As a result, it is necessary to know which camera produced the sRGB output and use a method or network tailored for that sensor to properly unprocess it. This appendix addresses this limitation by exploiting another camera image state that is not available as an output, but it is available inside the camera pipeline. In particular, cameras apply a colorimetric conversion step to convert the raw-RGB image to a device-independent space based on the CIE XYZ color space before they apply the nonlinear photo-finishing. Leveraging this canonical image state, we propose a deep learning framework, CIE XYZ Net, that can unprocess a nonlinear image back to the canonical CIE XYZ image. This image can then be processed by any low-level computer vision operator (e.g., defocus estimation) and re-rendered back to the nonlinear image. We demonstrate the usefulness of the CIE XYZ Net on several low-level vision tasks with a focus on the defocus estimation task; and show notable gains that can be

obtained by this processing framework. The work presented in this appendix is a part of our publication in the IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2021 [219].

D.1 Introduction

An image signal processor (ISP) onboard a camera processes the initial captured sensor image in a pipeline fashion, with routines being applied one after the other. The ISP used by consumer cameras performs operations as two distinct stages. First, a "front-end" stage applies linear operations, such as white balance and color adaptation, to convert the sensor-specific raw-RGB image to a device-independent color space (e.g., CIE XYZ or its wide-gamut representation, ProPhoto) [220]. The image states associated with the front-end process are called a *scene-referred* image because the image remains related directly to initial recorded sensor values related to the physical scene. Next, a "photo-finishing" stage is performed that applies nonlinear steps and local operators to produce a visually pleasing photograph. For example, selective color manipulation is often applied to enhance skin tone or make the overall colors more vivid, while local tone manipulation increases local contrast within the image. After the photo-finishing stage, the image is encoded in an output color space (e.g., sRGB, AdobeRGB, or Display P3). The image states associated with the photo-finishing process are referred to as *display-referred* as they are encoded for visual display. Cameras currently allow access only to either the minimally processed scene-referred image state (i.e., raw-RGB image) or the final display-referred image state (e.g., sRGB, AdobeRGB, or Display P3). Unfortunately, these two image states are not ideal for low-level computer vision tasks like defocus estimation.

The raw-RGB image state preserves the linear relationship of incident scene radiance. This linear image formation makes raw-RGB images suitable for a wide range of low-level computer vision tasks, such as image deblurring, image dehazing, image denoising, and various types of image enhancement [22, 222–224]. However, the drawback of raw-RGB is that the physical color filter arrays that make up the sensor's Bayer pattern are sensor-specific. This means raw-RGB values captured of the same scene but with different sensors are significantly different [225]. This often requires learning-based methods to be trained per sensor or camera make and model (e.g., [223, 226–229]).

The more common display-referred image state (in this appendix, assumed to be in

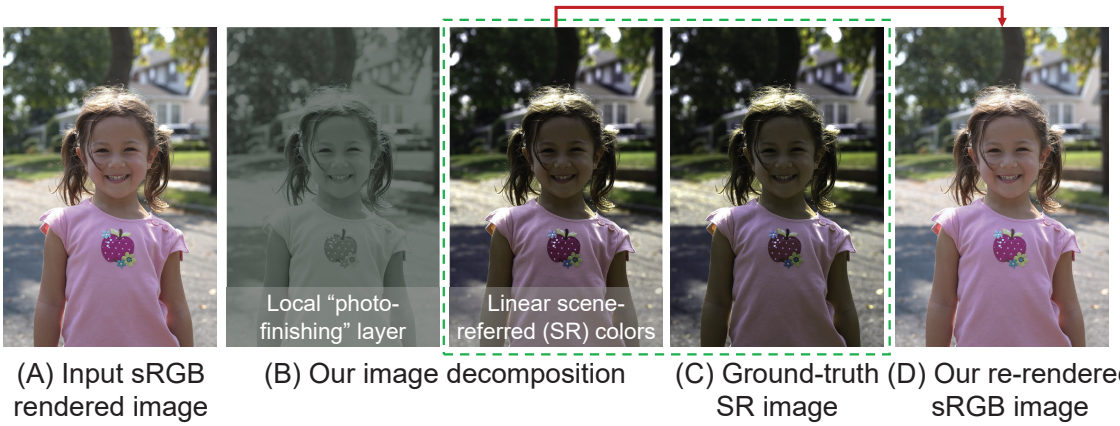


Figure D.1: We propose a cycle framework that can unprocess sRGB images back to the linear CIE XYZ color space and re-render the CIE XYZ images into the nonlinear sRGB color space. (A) The input camera-rendered sRGB image. (B) Our image decomposition (left: residual photo-finishing layer, right: scene-referred CIE XYZ reconstruction). (C) The ground-truth scene-referred CIE XYZ image. (D) Our re-rendering result from the reconstructed CIE XYZ image. To aid visualization, CIE XYZ images are scaled by a factor of two. Input image is taken from the MIT-Adobe FiveK dataset [221].

the sRGB color space) also has drawbacks. While this image state is the most widely used and is suitable for display, cameras apply their own proprietary photo-finishing to enhance the visual quality of the image. This means images captured of the same scene but using different camera models (and sometimes the same camera but with different settings) will produce images that have notably different sRGB values [220, 222, 230].

As previously discussed, the front-end processor of a typical camera ISP performs a colorimetric conversion to map the raw-RGB image to a standard perceptual colorspace—namely, CIE 1931 XYZ [220]. While there exists no formal image encoding for this image state, it is possible to convert existing raw-RGB images stored in digital negative (DNG) format to this intermediate state by applying a software camera ISP (e.g., [220, 231]). This provides a mechanism to standardize all images into a canonical linear scene-referred image state and is the impetus of our work.

In this appendix, we propose a method to decompose non-linear sRGB images into two parts: 1) a canonical linear scene-referred image state in the CIE XYZ color space and 2) a residual image layer that resembles additional non-linear and local photo-finishing

operations. Through such decomposition strategy, we learn a model that can accurately map back and forth between non-linear sRGB and linear CIE XYZ images. An example is shown in Figure D.1. Unlike raw-RGB, the CIE XYZ color space is *device-independent*, and as a result, helps with model generalization. Furthermore, CIE XYZ images can be encoded as standard three-channel images that can be easily handled by existing computer vision frameworks. We show that our proposed model maps images back to the CIE XYZ color space more accurately compared to alternative approaches. In addition, we validated the apprach on the defocus estimation task, to show that employing our proposed CIE XYZ model provides the performance boost anticipated from using linear images.

D.2 Background

In this section, we first discuss the camera imaging pipeline that is necessary in order to understand how and why we access the camera image once converted to the CIE XYZ color space. We then review various methods proposed for linearization of camera-rendered images.

D.2.1 Camera Imaging Pipeline

Images captured by a camera undergo a sequence of processes in order to transform an initial image obtained by the raw sensor data (raw-RGB image) into a visually perceivable color space (e.g., sRGB) and a suitable format for storage and display (e.g., JPEG) [119, 220, 232, 233]. A simplified depiction of the imaging pipeline is shown in Figure D.2. The pipeline processes can be divided into two main stages. The first stage is colorimetric processing that transforms the image into a linear color space (e.g., CIE XYZ or ProPhoto) that preserves the direct mapping between the recorded scene values and the image. The second stage in the camera pipeline is the camera-rendering or photo-finishing stage that applies nonlinear transformations (e.g., gamma compression), selective color manipulation, and locally varying processing (e.g., local tone mapping) that break the relation between the image and the scene. Using images in the linear stages of the imaging pipeline has been shown to be more effective for image restoration and enhancement tasks [220, 222, 223]. However, due to the complexity and proprietary nature of imaging pipelines on-board cameras, it is hard to obtain colorimetric images from cameras without the effort of saving raw sensor data or inverting the imaging pipeline stages [222, 223].

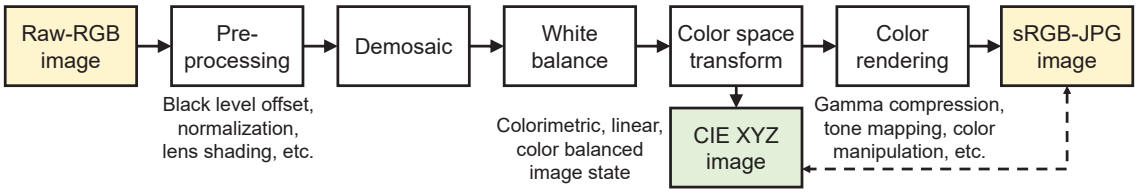


Figure D.2: A simplified depiction of a camera imaging pipeline, adapted from [119, 220, 232, 233]. Our method allows mapping back and forth between the common nonlinear sRGB image and to the colorimetric, linear, color-balanced CIE XYZ image state for computer vision tasks.

D.2.2 Camera-Rendered Image Linearization

To obtain a linear image from its camera-rendered version, we need to reverse the nonlinear camera-rendering stage in the pipeline. Many methods have been proposed to model a parametric relationship that maps from the camera-rendered image (i.e., sRGB image) back to its raw-RGB version (e.g., [222]). However, raw-RGB space is camera-dependent and requires having a separate model per camera. Other approaches involve simple linearization by inverting the global tone mapping and the gamma compression followed by applying a linearization matrix to obtain a linear sRGB or CIE XYZ image [223]. Such approaches are too simple and do not account for the local processing or dynamic range adjustments. Unlike prior approaches, instead of trying only to obtain a linear image, our approach is to decompose the nonlinear image into globally processed and locally processed layers. The locally processed layer represents local color processing, such as local tone mapping. Then, we learn a global mapping from the globally processed image to the linear image. Another line of research targeting the problem of image linearization is radiometric calibration [234, 235]. Unlike our approach, radiometric calibration methods do not target a specific, well-defined color space, and do not address the problem of local processing.

D.3 Our Framework

This section describes our overall framework, including network architecture, dataset generation, and training details.

D.3.1 Formulation

Inside a camera imaging pipeline, a raw-RGB image $\mathbf{x}_{\text{raw}} \in \mathbb{R}^{h \times w}$ undergoes a sequence of processing stages to be transformed to the final output sRGB image $\mathbf{x}_{\text{srgb}} \in \mathbb{R}^{h \times w \times 3}$, where h and w represent the image height and width, respectively. As mentioned earlier, the raw-RGB image $\mathbf{x}_{\text{raw}}$ is in a camera-dependent color space that is linear with respect to scene light irradiance falling on the sensor. One of the early steps in the camera processing pipeline is to convert the camera-dependent color space to a device-independent color space—namely, CIE XYZ. Based on this observation, instead of modeling the whole pipeline back to the raw-RGB image, we choose to model an intermediate representation of the image in the CIE XYZ color space $\mathbf{x}_{\text{xyz}} \in \mathbb{R}^{h \times w \times 3}$ that is still linear with respect to scene irradiance, but is in a canonical color space. We are interested in the on-camera rendering procedures that map the CIE XYZ images into the final display-referred (i.e., photo-finished) sRGB color space. This operation can be described as

$$\mathbf{x}_{\text{srgb}} = \mathcal{F}(\mathbf{x}_{\text{xyz}}). \tag{D.1}$$

In our method, instead of relying on a single function to model the pipeline stages between sRGB and CIE XYZ, we decompose this mapping into two parts: 1) *global processing*, denoted collectively as $\mathcal{F}_{\text{glob}}(\cdot)$, that is globally applied to all image pixels and 2) *local processing*, denoted collectively as $\mathcal{F}_{\text{loc}}(\cdot)$, that represents local photo-finishing operations, such as local tone mapping and selective color adjustments. Such design is largely motivated by the fact that actual camera image processing pipelines (ISPs) perform both global and local image processing. Global processing can be easily modeled by a polynomial color mapping, and hence, we train a CNN to estimate such polynomial function coefficients. Local processing is more challenging to model, and hence, we chose to model it as a residual fully-convolutional neural network. Breaking the process into two parts, global and local, has the added advantage that it enables image enhancement methods to selectively manipulate either part independently, and improves the performance of various photo-finishing tasks.

Our forward pipeline from $\mathbf{x}_{\text{xyz}}$ to $\mathbf{x}_{\text{srgb}}$ can be represented as a cascade of the global and the local processes. The global processing stage is represented as

$$\mathbf{M}_{\text{fwd}} = \mathcal{F}_{\text{glob}}(\mathbf{x}_{\text{xyz}}), \tag{D.2}$$

$$\mathbf{x}_{\text{glob}} = \psi\left(\mathbf{M}_{\text{fwd}}\ \phi(\mathbf{x}_{\text{xyz}})\right), \tag{D.3}$$

where $\mathbf{M}_{\text{fwd}} \in \mathbb{R}^{3\times 6}$ is a global transformation matrix and $\mathbf{x}_{\text{glob}}$ is the globally processed image layer. The operator $\phi(\cdot)$ reshapes the image to be $6 \times n$ where n is the number of pixels in the image and each pixel is transformed from three to six dimensions: $[R, G, B] \rightarrow [R, G, B, R^2, G^2, B^2]$, while the operator ψ reshapes the image from $3 \times n$ back to $h \times w \times 3$. We chose $\mathbf{M}_{\text{fwd}}$ to be nonlinear to capture global color processing operations, such as gamma compression.

As most consumer cameras locally process the captured scene-referred images to improve the quality of final rendered images [236], such global color processing may not be able to effectively model the function $\mathcal{F}$. To that end, we use a residual learning mechanism where we model the residual layer $\mathbf{x}_{\text{res}}$ between the locally and globally processed layers of the image as follows:

$$\mathbf{x}_{\text{res}} = \mathcal{F}_{\text{loc}}(\mathbf{x}_{\text{glob}}), \tag{D.4}$$

$$\mathbf{x}_{\text{srgb}} = \mathbf{x}_{\text{glob}} + \mathbf{x}_{\text{res}}. \tag{D.5}$$

Now, the decomposition process applies the inverse process of Equations D.2 – D.5 as follows:

$$\mathbf{x}_{\text{res}} = \mathcal{G}_{\text{loc}}(\mathbf{x}_{\text{srgb}}), \tag{D.6}$$

$$\mathbf{x}_{\text{glob}} = \mathbf{x}_{\text{srgb}} - \mathbf{x}_{\text{res}}, \tag{D.7}$$

$$\mathbf{M}_{\text{inv}} = \mathcal{G}_{\text{glob}}(\mathbf{x}_{\text{glob}}), \tag{D.8}$$

$$\mathbf{x}_{\text{xyz}} = \psi\left(\mathbf{M}_{\text{inv}}\ \phi(\mathbf{x}_{\text{glob}})\right), \tag{D.9}$$

where $\mathcal{G}_{\text{loc}}(\cdot)$ represents the inverse of residual local processing layer and $\mathcal{G}_{\text{glob}}(\cdot)$ is constrained to produce a global transformation matrix $\mathbf{M}_{\text{inv}} \in \mathbb{R}^{3\times 6}$ that represents the inverse global processing stage.

Our ultimate goal is to allow the manipulation of the reconstructed CIE XYZ image by arbitrary image restoration/enhancement algorithms between the inverse and forward pipeline stages (see Figure D.3). It is, however, non-trivial to infer the inverse functions $\mathcal{G}_{\text{loc}}^{-1}(\cdot)$ and $\mathcal{G}_{\text{glob}}^{-1}(\cdot)$ to render back the reconstructed image, as its values may be changed by the image restoration or enhancement algorithms. To that end, we model each of $\mathcal{F}_{\text{glob}}(\cdot)$, $\mathcal{F}_{\text{loc}}(\cdot)$, $\mathcal{G}_{\text{glob}}(\cdot)$, and $\mathcal{G}_{\text{loc}}(\cdot)$ by a neural network.

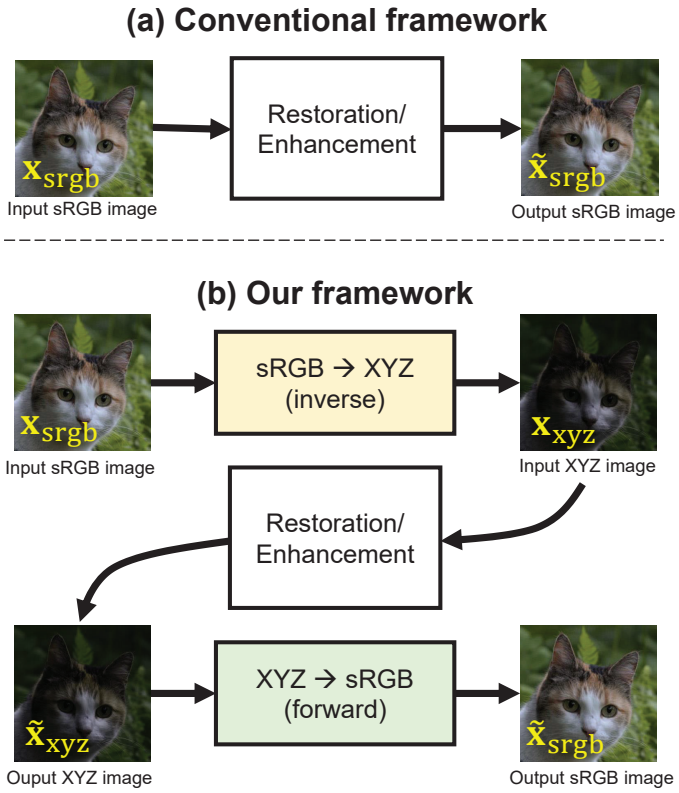


Figure D.3: An illustration of using our inverse and forward image processing pipelines in an sRGB image restoration/enhancement framework.

D.3.2 Network Design

Imitating this division of the camera imaging pipeline, we build our network architecture to include two sub-networks for modeling both the global and local processing parts for the forward and inverse directions of the imaging pipeline. As shown in Figure D.4, we start with the inverse pipeline where the first part is a fully-convolutional neural network (CNN) that models the local processing applied to an input non-linear image (i.e., sRGB image) by predicting the residual image $\mathbf{x}_{\text{res}}$ (Equation D.6). Once the local processing layer is predicted, it can be subtracted from the input image $\mathbf{x}_{\text{srgb}}$ to get the globally processed image $\mathbf{x}_{\text{glob}}$ (Equation D.7). Then, $\mathbf{x}_{\text{glob}}$ is fed to another sub-network that predicts a global transformation $\mathbf{M}_{\text{inv}}$ that inverts $\mathbf{x}_{\text{glob}}$ back to the linear CIE XYZ image $\mathbf{x}_{\text{xyz}}$ (Equation D.9). With this inverse pipeline, we decompose the input image $\mathbf{x}_{\text{srgb}}$ into two image layers, $\mathbf{x}_{\text{res}}$ and $\mathbf{x}_{\text{glob}}$, which represent local and global processing, respectively, and

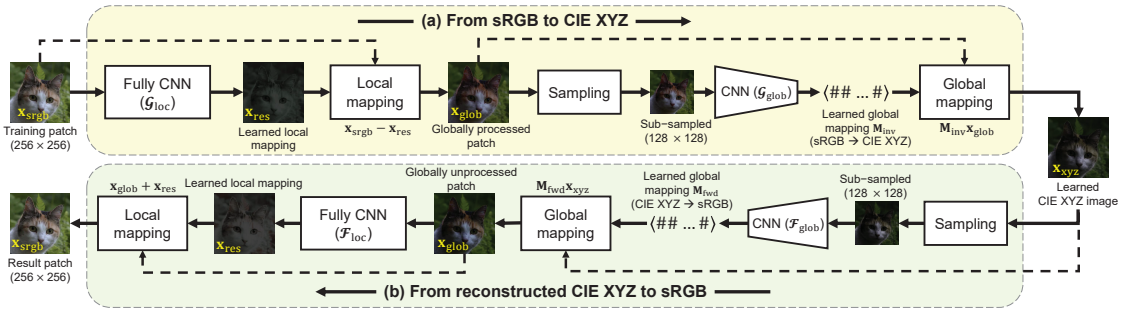


Figure D.4: Our CIE XYZ image pipeline. The upper part is the inverse pipeline that unprocesses an sRGB image into a CIE XYZ image. The lower part is the forward pipeline that processes a CIE XYZ image into its equivalent sRGB image. The full framework is trainable end-to-end. The CIE XYZ images are scaled 2x to aid visualization.

finally output the linear CIE XYZ image $\mathbf{x}_{\text{xyz}}$.

As discussed in Section D.1, there are computer vision tasks, such as image restoration, that are best processed in a linear image state. A use case of the framework is to convert the input image *imxyz*, process the *imxyz* image, and then render the image back. In this scenario, after decomposing an image and applying an image restoration task to the linear XYZ image, we now need to merge these image layers back to produce the fully processed sRGB image. To model this forward pass of our pipeline, as shown in Figure D.4, we use two sub-networks. The first sub-network predicts a global transformation $\mathbf{M}_{\text{fwd}}$ that maps $\mathbf{x}_{\text{xyz}}$ to $\mathbf{x}_{\text{glob}}$ (Equation D.3). The second sub-network predicts the residual local processing $\mathbf{x}_{\text{res}}$ that needs to be applied to $\mathbf{x}_{\text{glob}}$ to obtain the final sRGB image $\mathbf{x}_{\text{srgb}}$ (Equation D.5). This framework is illustrated in Figure D.3 and compared to the conventional way of directly processing the sRGB image.

In order to allow the networks $\mathcal{G}_{\text{glob}}(\cdot)$ and $\mathcal{G}_{\text{loc}}(\cdot)$ to separate the globally and locally processed image layer without having ground truth for both $\mathbf{x}_{\text{glob}}$ and $\mathbf{x}_{\text{res}}$, we apply a scaling factor to the output of the local processing networks $\mathcal{G}_{\text{glob}}(\cdot)$, in both inverse and forward passes, such that the values of $\mathbf{x}_{\text{res}}$ are much smaller than $\mathbf{x}_{\text{glob}}$. In our experiments, we set this scaling factor to 0.25. Figure D.5 shows an example of the output of each sub-network. It is challenging to evaluate our global and local processing modules separately; mainly because camera ISP global and local processing modules are typically proprietary and not accessible, and hence, we cannot obtain ground truth data for evaluation.

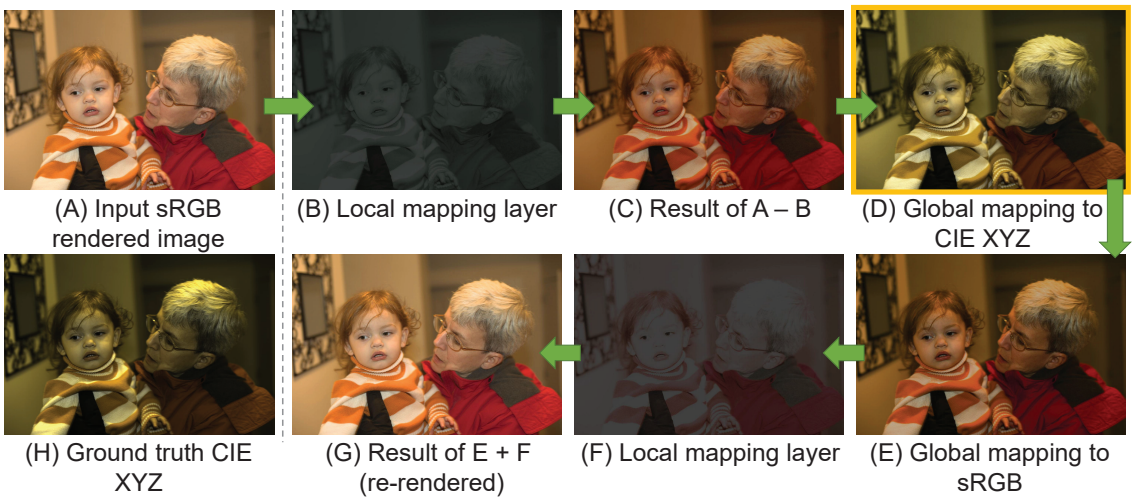


Figure D.5: Our inverse pipeline decomposites a given camera-rendered sRGB image into a local processed layer and the corresponding CIE XYZ image, while our forward pipeline maps the reconstructed CIE XYZ image to the sRGB color space in an inverse way of our decomposition. The shown image is taken from our testing set. To aid visualization, CIE XYZ images are scaled by a factor of two.

D.3.3 Loss Function

The objective of the whole network is to minimize the mean absolute error (MAE): 1) between the predicted XYZ image $\hat{\mathbf{x}}_{\text{xyz}}$ and its ground truth $\mathbf{x}^*_{\text{xyz}}$ in the inverse pipeline and 2) between the predicted sRGB image $\hat{\mathbf{x}}_{\text{srgb}}$ and its ground truth $\mathbf{x}^*_{\text{srgb}}$ in the forward pipeline, as follows:

$$\lambda \left|\hat{\mathbf{x}}_{\text{xyz}} - \mathbf{x}^*_{\text{xyz}}\right| + \left|\hat{\mathbf{x}}_{\text{srgb}} - \mathbf{x}^*_{\text{srgb}}\right|, \tag{D.10}$$

where λ is a weighting factor that we use to deal with the fact that XYZ images generally have lower intensity compared to sRGB images; so this weight can balance the learning behavior between the forward and inverse pipelines. In our experiments, we set $\lambda = 1.5$.

D.3.4 Sub-Networks Architecture

Our local processing sub-networks ($\mathcal{F}_{\text{loc}}$ and $\mathcal{G}_{\text{loc}}$) each consist of 15 blocks of 3×3 convolutional (conv)–LReLU layers. Each conv layer has 32 output channels, with stride of 1 and padding of 1. The last layer of these sub-networks has a single conv layer with three output channels, followed by a tanh operator. As our global processing sub-networks are

not fully convolutional, we use a fixed size of input by introducing a differentiable sub-sampling module that uniformly subsamples 128×128 color values of the processed image by the previous sub-network. Our global sub-network includes five blocks of 3×3 conv–LReLU–2×2 max pooling layers. The conv layers have stride and padding of 1, while the max pooling layers have a stride factor of 2 with no padding. Then, we added a fully connected layer with 1024 output neurons, followed by a dropout layer with a factor of 0.5. The last layer of our global sub-network has a fully connected layer with 18 output neurons to formulate our 3×6 polynomial mapping function. Our entire framework is a light-weight model with a total of 2,697,578 learnable parameters ($\sim$11MB of memory) for both sRGB-to-XYZ and XYZ-to-sRGB models, and it is fully differentiable for end-to-end training.

D.3.5 Dataset

To train our proposed model, we need a dataset of sRGB images with their corresponding linear images in the CIE XYZ color space. To do so, we start from raw-RGB images taken from the MIT-Adobe FiveK [221]. We then process the raw-RGB images twice to obtain both the sRGB and XYZ versions of each image. For processing raw-RGB images into the XYZ color space, we used the camera pipeline from [231]. This pipeline provides an access to the CIE XYZ values after processing the sensor raw-RGB using the color space transformation (CST) matrices provided with the raw-RGB image. To obtain the camera-like sRGB images, we used the Adobe Camera RAW software development kit (SDK), which accurately emulates the nonlinearity applied by consumer cameras [237].

The MIT-Adobe FiveK [221] dataset contains images captured with different cameras. As a result, the CIE XYZ and sRGB images are rendered with different processing profiles according to the metadata from each camera.

Our method's CIE XYZ fidelity evaluations are tied to the used camera models in the MIT-Adobe FiveK [221]. However, this does not take away from the advantages that the standard canonical CIE XYZ space offers over other linear spaces as a target space for our supervision learning. Our assumption is that by training on images rendered by a broad range of ISP emulations for different camera models, we can learn to unprocess generic processing applied by most cameras in order to achieve a better linearization. Our dataset includes $\sim$1,200 pairs of sRGB and camera CIE XYZ images. Our dataset will be publicly

available upon acceptance.

D.3.6 Training

We divided our dataset into a training set of 971 pairs, a validation set of 50 pairs, and a testing set of 244 pairs. We trained our framework in an end-to-end manner on patches of size 256×256 pixels randomly extracted from our training set, with a mini-batch of size 4. We applied random geometric augmentation (i.e., scaling and reflection) to the extracted patches.

Our framework was trained in an end-to-end manner for 300 epochs using Adam optimizer [90] with gradient decay factor $\beta_1 = 0.9$ and squared gradient decay factor $\beta_2 = 0.999$. We used a learning rate of 10^{-4} with a drop factor of 0.5 every 75 epochs. We added an L_2 regularization with a weight of $\lambda_{\text{reg}} = 10^{-3}$ to our loss in Equation D.10 to avoid overfitting.

D.4 Experimental Results

In this section, we first validate the effectiveness of our proposed model in mapping from camera-rendered sRGB images to CIE XYZ, and processing CIE XYZ images back to sRGB. Next, we demonstrate our method's utility on defocus estimation that assume a linear relationship between the scene radiance and the recorded pixel intensity.

D.4.1 From Camera-Rendered sRGB to CIE XYZ, and Back

We first verify our network's ability to unprocess sRGB images to CIE XYZ. We also demonstrate our ability to reconstruct from CIE XYZ back to sRGB. We test our mapping to sRGB using our reconstructed CIE XYZ results as a starting point, and also using the ground-truth CIE XYZ images.

We compared our method with three existing methods. First, we compare with the *standard CIE XYZ mapping* [238, 239], which applies a simple 2.2 gamma tone curve. Second, we compare with the recent *unprocessing technique (UPI)* from [223]. The UPI unprocessing technique is non-trainable and inverts the camera ISP, step-by-step, through a series of transformations, such as gamma expansion and inverting color correction matrices. This unprocessing module is then used to generate realistic training data for the

Table D.1: Results (in terms of PSNR) of camera-rendered sRGB ↔ CIE XYZ mapping. We compare our results against the standard XYZ mapping (the 2.2 gamma tone curve) [238, 239], the recent unprocessing technique (UPI) [223], and CycleISP [224]. Average PSNR (dB) results are reported on 244 unseen testing pairs (camera-rendered sRGB and corresponding CIE XYZ images) from the MIT-Adobe FiveK dataset [221]. We show results of mapping from both reconstructed (Rec.) CIE XYZ images and ground truth (GT) CIE XYZ images to the corresponding camera-rendered sRGB images. Highest PSNR values are shown in bold.

Method	**sRGB → XYZ**				**Rec. XYZ → sRGB**				**GT XYZ → sRGB**			
	Avg.	**Q1**	**Q2**	**Q3**	**Avg.**	**Q1**	**Q2**	**Q3**	**Avg.**	**Q1**	**Q2**	**Q3**
Standard [238, 239]	21.84	16.88	20.91	25.24	-	-	-	-	22.22	19.19	21.79	24.37
Unprocessing [223]	22.19	19.31	22.12	24.75	37.72	37.78	40.56	41.88	18.04	15.67	17.79	20.02
CycleISP [224]	28.29	23.63	28.08	31.98	34.78	30.60	34.20	37.22	20.91	18.36	21.42	24.31
Ours	**29.66**	**23.77**	**29.57**	**34.71**	**43.82**	**41.43**	**43.94**	**46.58**	**27.44**	**23.57**	**28.32**	**30.88**

task of image denoising. It is only the denoising module of [223] that is a CNN, and that is trainable. For a fair comparison, we compare our results with results of UPI obtained at the CIE XYZ stage. Third, we compare with CycleISP [224], a recent network architecture that aims at simulating the camera ISP mapping between raw and sRGB stages. Since CycleISP is targeting a camera-specific sRGB-to-raw mapping, we had to introduce some slight modifications to their architecture to make it suitable for our objective, i.e., mapping between sRGB and CIE XYZ. In particular, we omitted their noise injection and color correction modules and modified their RGB2RAW and RAW2RGB networks to have 3-channel inputs and outputs. We retrained the CycleISP with the same training settings used to train our model for a fair comparison.

Table D.1 shows peak signal-to-noise ratio (PSNR) results averaged over 244 unseen testing images from the MIT-Adobe FiveK dataset [221]. The terms Q1, Q2, and Q3 refer to the first, second (median), and third quantile, respectively, of the PSNR values obtained by each method. For the standard XYZ, the results of mapping from the reconstructed CIE XYZ images back to sRGB are not reported because standard XYZ uses an invertible transform. The sRGB reconstruction error from the UPI model [223] is high due to the fact that the tone mapping is not perfectly invertible. It can be observed from the results that we outperform all competing methods by a sound margin. Qualitative comparisons are provided in Figure 7.5.

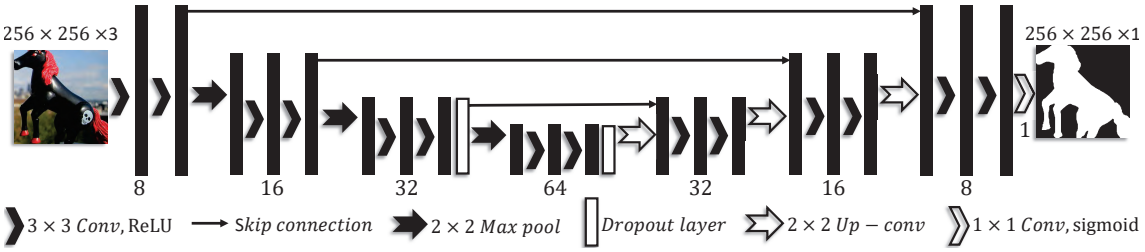


Figure D.6: We used a light-weight U-Net-like architecture for the task of defocus blur detection. The size of the input and output layers is shown above the image patches. The number of output filters is shown under the convolution operations.

As shown in Table D.1, the mapping to sRGB from reconstructed CIE XYZ is better than mapping from ground-truth CIE XYZ. For our method, this behavior is expected because the forward model is trained on the reconstructed CIE XYZ, not the ground truth. Also, for the UPI method, as it is based on matrix inversion, the mapping from the reconstructed CIE XYZ makes the transformation more accurate than mapping from the ground truth.

D.4.2 Defocus Estimation

Defocus detection is the problem of detecting the image pixels that are out of focus. This problem is important to many computer vision tasks, such as non-blind defocus deblurring, image refocusing, depth estimation, and 3D reconstruction. The methods targeting this problem are well established [60–65]. In particular, defocus detection methods output a binary mask of a given input image such that the out-of-focus pixels are zeros and the rest are ones.

We examine the advantage of detecting out-of-focus pixels in our linearized CIE XYZ color space compared to other spaces—namely, non-linear sRGB and standard linearized CIE XYZ. To this aim, we introduce a light UNet-inspired architecture as shown in Figure D.6. We train three models using image patches from three different color spaces following the same training procedure. We use the well-known blur detection dataset [61] to train and test the models.

We trained the models with image patches of size 256×256. We adopted He's weight initialization [100] and used the Adam optimizer [90] to train our model. The initial learning rate is set to 10^{-4}, which is decreased by half every 30 epochs. We train the

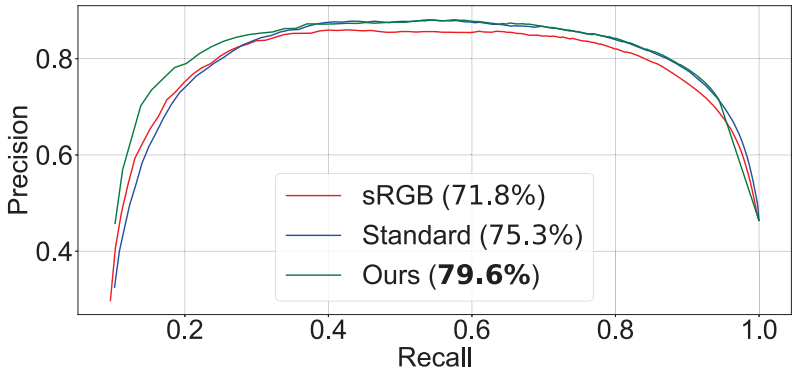


Figure D.7: The precision-recall (PR) comparison of training light UNet on data from three different color spaces: sRGB, standard linearized CIE XYZ [238, 239], and our linearized CIE XYZ. The average accuracy for each model is shown in the plot's legend. Our linear space achieves the best PR curve and the highest accuracy.

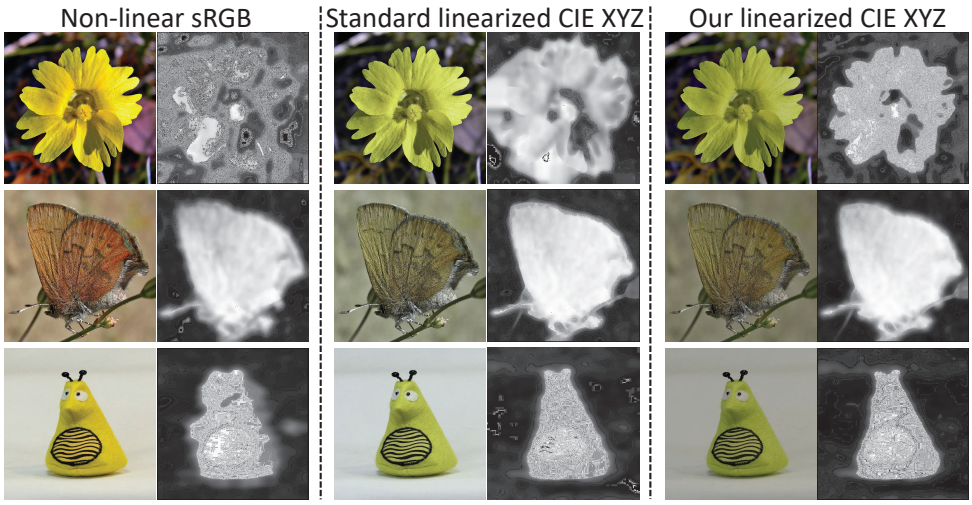


Figure D.8: Qualitative results of three light UNets trained on three different color spaces for the task of defocus map estimation. Training the light UNet using our linearized CIE XYZ images gives better visual results as shown in the third column. The CIE XYZ input images are gamma corrected for better visualization.

model with mini-batches of size 12 using the mean squared error (MSE) loss between the output and the ground truth. During the training phase, we set the dropout rate to 0.5. We found that the model converges after 60 epochs.

In the plot of Figure D.7, we present the precision-recall comparison along with the

pixel binary classification accuracy. In addition to the quantitative results, qualitative results are presented in Figure D.8. The results in Figure D.7 and Figure D.8 demonstrate that our linearization is a better color space to perform defocus blur detection compared to other color spaces.

D.5 Concluding remarks

In this appendix we have presented our method and DNN model that can map back and forth between non-linear sRGB and linear CIE XYZ images more accurately compared to alternative approaches. Our method is based on learning a decomposition of sRGB images into a globally processed and locally processed image layers. The learned globally processed image layer is then used to learn a mapping to the device independent CIE XYZ color space. We have provided experiments targeting accurate mapping and defocus estimation to show that our proposed model provides the performance boost anticipated from using linear images.

Bibliography

[1] Abraham Savitzky and Marcel JE Golay, "Smoothing and differentiation of data by simplified least squares procedures," *Analytical chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

[2] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia, "Scale-recurrent network for deep image deblurring," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[3] Abdullah Abuolaim and Michael S Brown, "Defocus deblurring using dual-pixel data," in *European Conference on Computer Vision (ECCV)*, 2020.

[4] Abhijith Punnappurath, Abdullah Abuolaim, Mahmoud Afifi, and Michael S Brown, "Modeling defocus-disparity in dual-pixel sensors," in *IEEE International Conference on Computational Photography (ICCP)*, 2020.

[5] Abhijith Punnappurath and Michael S Brown, "Reflection removal using a dual-pixel sensor," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[6] Filiberto Chiabrando, Roberto Chiabrando, Dario Piatti, and Fulvio Rinaudo, "Sensors for 3d imaging: Metric evaluation and calibration of a ccd/cmos time-of-flight camera," *Sensors*, vol. 9, no. 12, pp. 10080–10096, 2009.

[7] Bernhard Buttgen and Peter Seitz, "Robust optical time-of-flight range imaging based on smart pixel structures," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 6, pp. 1512–1525, 2008.

[8] Keiji Ohsawa, "Focus detecting device and method of operation," 1996, US Patent 5,530,513.

[9] Daisuke Inoue and Hidekazu Takahashi, "Focus detecting device and camera system using the same device," 2009, US Patent 7,577,349.

[10] Przemysław Śliwiński and Paweł Wachel, "A simple model for on-sensor phase-detection autofocusing algorithm," *Journal of Computer and Communications*, vol. 1, no. 06, pp. 11, 2013.

[11] Jinbeum Jang, Yoonjong Yoo, Jongheon Kim, and Joonki Paik, "Sensor-based auto-focusing system using multi-scale feature extraction and phase correlation matching," *Sensors*, vol. 15, no. 3, pp. 5747–5762, 2015.

[12] Rudi Chen and Peter van Beek, "Improving the accuracy and low-light performance of contrast-based autofocus using supervised machine learning," *Pattern Recognition Letters*, vol. 56, pp. 30–37, 2015.

[13] Kazushige Ooi, Keiji Izumi, Mitsuyuki Nozaki, and Ikuya Takeda, "An advanced autofocus system for video camera using quasi condition reasoning," *IEEE Transactions on Consumer Electronics*, vol. 36, no. 3, pp. 526–530, 1990.

[14] Abdullah Abuolaim, Abhijith Punnappurath, and Michael S Brown, "Revisiting autofocus for smartphone cameras," in *European Conference on Computer Vision (ECCV)*, 2018.

[15] Jinshan Pan, Haoran Bai, and Jinhui Tang, "Cascaded deep video deblurring using temporal sharpness prior," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[16] Jochen Gast and Stefan Roth, "Deep video deblurring: The devil is in the details," in *International Conference on Computer Vision Workshops (ICCVW)*, 2019.

[17] Seungjun Nah, Sanghyun Son, and Kyoung Mu Lee, "Recurrent neural networks with intra-frame iterations for video deblurring," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[18] Shangchen Zhou, Jiawei Zhang, Jinshan Pan, Haozhe Xie, Wangmeng Zuo, and Jimmy Ren, "Spatio-temporal filter adaptive network for video deblurring," in *International Conference on Computer Vision (ICCV)*, 2019.

[19] Shangchen Zhou, Jiawei Zhang, Wangmeng Zuo, Haozhe Xie, Jinshan Pan, and Jimmy S Ren, "Davanet: Stereo deblurring with view aggregation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[20] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[21] Shuochen Su, Mauricio Delbracio, Jue Wang, Guillermo Sapiro, Wolfgang Heidrich, and Oliver Wang, "Deep video deblurring for hand-held cameras," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[22] Yu-Wing Tai, Xiaogang Chen, Sunyeong Kim, Seon Joo Kim, Feng Li, Jie Yang, Jingyi Yu, Yasuyuki Matsushita, and Michael S Brown, "Nonlinear camera response functions and image deblurring: Theoretical analysis and practice," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 10, pp. 2498–2512, 2013.

[23] Abdullah Abuolaim and Michael Brown, "Online lens motion smoothing for video autofocus," in *Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[24] W Machado and et al., "Canon LC1290A (Die Markings) 20.2 Mp, 4.1 μm pixel size dual pixel CMOS AF APS-C CMOS image sensor from the canon EOS-70D(W) DSLR camera imager process review," 2013.

[25] Rahul Garg, Neal Wadhwa, Sameer Ansari, and Jonathan T Barron, "Learning single camera depth estimation using dual-pixels," in *International Conference on Computer Vision (ICCV)*, 2019.

[26] Yinda Zhang, Neal Wadhwa, Sergio Orts-Escolano, Christian Häne, Sean Fanello, and Rahul Garg, "Du2net: Learning depth estimation from dual-cameras and dual-pixels," *European Conference on Computer Vision (ECCV)*, 2020.

[27] Neal Wadhwa, Rahul Garg, David E Jacobs, Bryan E Feldman, Nori Kanazawa, Robert Carroll, Yair Movshovitz-Attias, Jonathan T Barron, Yael Pritch, and Marc

Levoy, "Synthetic depth-of-field with a single-camera mobile phone," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 64, 2018.

[28] Thuy Tuong Nguyen and Jae Wook Jeon, "Camera auto-exposing and auto-focusing for edge-related applications using a particle filter," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[29] Loren Shih, "Autofocus survey: a comparison of algorithms," in *Digital Photography III*. International Society for Optics and Photonics, 2007, vol. 6502, p. 0B.

[30] Sang-Yong Lee, Yogendera Kumar, Ji-Man Cho, Sang-Won Lee, and Soo-Won Kim, "Enhanced autofocus algorithm using robust focus measure and fuzzy reasoning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 9, pp. 1237–1246, 2008.

[31] Jaehwan Jeon, Jinhee Lee, and Joonki Paik, "Robust focus measure for unsupervised auto-focusing based on optimum discrete cosine transform coefficients," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 1, 2011.

[32] Matej Kristan, Janez Perš, Matej Perše, and Stanislav Kovačič, "A bayes-spectral-entropy-based measure of camera focus using a discrete cosine transform," *Pattern Recognition Letters*, vol. 27, no. 13, pp. 1431–1439, 2006.

[33] Hashim Mir, Peter Xu, and Peter Van Beek, "An extensive empirical evaluation of focus measures for digital photography," in *Digital Photography X*. International Society for Optics and Photonics, 2014, vol. 9023, p. 0I.

[34] William B Pennebaker and Joan L Mitchell, *JPEG: Still image data compression standard*, Springer Science & Business Media, 1992.

[35] Jie He, Rongzhen Zhou, and Zhiliang Hong, "Modified fast climbing search auto-focus algorithm with adaptive step size searching technique for digital camera," *IEEE transactions on Consumer Electronics*, vol. 49, no. 2, pp. 257–262, 2003.

[36] Quoc Kien Vuong and Jeong-won Lee, "Initial direction and speed decision system for auto focus based on blur detection," in *IEEE International on Conference Consumer Electronics*, 2013.

[37] Naoto Nakahara, "Passive autofocus system for a camera," 2006, US Patent 7,058,294.

[38] Kang-Sun Choi, Jun-Suk Lee, and Sung-Jae Ko, "New autofocusing technique using the frequency selective weighted median filter for video cameras," *IEEE Transactions on Consumer Electronics*, vol. 45, no. 3, pp. 820–827, 1999.

[39] M Gamadia and N Kehtarnavaz, "A real-time continuous automatic focus algorithm for digital cameras," in *IEEE Southwest Symposium on Image Analysis and Interpretation*, 2006.

[40] June-Sok Lee, You-Young Jung, Byung-Soo Kim, and Sung-Jea Ko, "An advanced video camera system with robust af, ae, and awb control," *IEEE Transactions on Consumer Electronics*, vol. 47, no. 3, pp. 694–699, 2001.

[41] Peter M Atkinson, C Jeganathan, Jadu Dash, and Clement Atzberger, "Inter-comparison of four models for smoothing satellite sensor time-series data to estimate vegetation phenology," *Remote Sensing of Environment*, vol. 123, pp. 400–417, 2012.

[42] Kirsten M de Beurs and Geoffrey M Henebry, "Spatio-temporal statistical methods for modelling land surface phenology," in *Phenological research*. 2010.

[43] Jeroen Jansze, "Time series machine learning technique with application to barcelona metro station energy minimization," M.S. thesis, 2013, University Utrecht.

[44] Robert De Levie, *Advanced Excel for scientific data analysis*, Book. Oxford University Press, 2004.

[45] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[46] Alex Graves and Jürgen Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[47] Qing Sun, Stefan Lee, and Dhruv Batra, "Bidirectional beam search: Forward-backward inference in neural sequence models for fill-in-the-blank image captioning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[48] Jingwen Wang, Wenhao Jiang, Lin Ma, Wei Liu, and Yong Xu, "Bidirectional attentive fusion with context gating for dense video captioning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[49] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[50] Jun Liu, Gang Wang, Ping Hu, Ling-Yu Duan, and Alex C Kot, "Global context-aware attention lstm networks for 3d action recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[51] Wentao Zhu, Cuiling Lan, Junliang Xing, Wenjun Zeng, Yanghao Li, Li Shen, Xiaohui Xie, et al., "Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks.," in *AAAI*, 2016.

[52] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *IEEE International Conference on Acoustics, speech and signal processing*, 2013.

[53] Andrej Karpathy and Li Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[54] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," in *IEEE Press. A Field Guide to Dynamical Recurrent Neural Networks*, 2001.

[55] Rudolph Emil Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[56] Pierre Del Moral, "Non-linear filtering: interacting particle resolution," *Markov processes and related fields*, vol. 2, no. 4, pp. 555–581, 1996.

[57] Pierre Del Moral et al., "Measure-valued processes and interacting particle systems. application to nonlinear filtering problems," *The Annals of Applied Probability*, vol. 8, no. 2, pp. 438–495, 1998.

[58] Pierre Moral, *Feynman-Kac Formulae*, Springer. Series: Probability and Applications, 2004.

[59] Pierre Del Moral, Arnaud Doucet, Ajay Jasra, et al., "On adaptive resampling strategies for sequential Monte Carlo methods," *Bernoulli*, vol. 18, no. 1, pp. 252–278, 2012.

[60] S Alireza Golestaneh and Lina J Karam, "Spatially-varying blur detection based on multiscale fused and sorted transform coefficients of gradient magnitudes.," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[61] Jianping Shi, Li Xu, and Jiaya Jia, "Discriminative blur detection features," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[62] Chang Tang, Xinzhong Zhu, Xinwang Liu, Lizhe Wang, and Albert Zomaya, "Defusionnet: Defocus blur detection via recurrently fusing and refining multi-scale deep features," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[63] Xin Yi and Mark Eramian, "Lbp-based segmentation of defocus blur," *IEEE Transactions on Image Processing (TIP)*, vol. 25, no. 4, pp. 1626–1638, 2016.

[64] Wenda Zhao, Fan Zhao, Dong Wang, and Huchuan Lu, "Defocus blur detection via multi-stream bottom-top-bottom fully convolutional network," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[65] Wenda Zhao, Bowen Zheng, Qiuhua Lin, and Huchuan Lu, "Enhancing diversity of defocus blur detectors via cross-ensemble network," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[66] Laurent D'Andrès, Jordi Salvador, Axel Kochale, and Sabine Süsstrunk, "Non-parametric blur map regression for depth of field extension," *IEEE Transactions on Image Processing (TIP)*, vol. 25, no. 4, pp. 1660–1673, 2016.

[67] Ali Karaali and Claudio Rosito Jung, "Edge-based defocus blur estimation with adaptive scale selection," *IEEE Transactions on Image Processing (TIP)*, vol. 27, no. 3, pp. 1126–1137, 2017.

[68] Junyong Lee, Sungkil Lee, Sunghyun Cho, and Seungyong Lee, "Deep defocus map estimation using domain adaptation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[69] Jinsun Park, Yu-Wing Tai, Donghyeon Cho, and In So Kweon, "A unified approach of multi-scale deep and hand-crafted features for defocus estimation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[70] Jianping Shi, Li Xu, and Jiaya Jia, "Just noticeable defocus blur detection and estimation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[71] Anat Levin, Yair Weiss, Fredo Durand, and William T Freeman, "Understanding blind deconvolution algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 33, no. 12, pp. 2354–2367, 2011.

[72] Huixuan Tang and Kiriakos N Kutulakos, "Utilizing optical aberrations for extended-depth-of-field panoramas," in *ACCV*, 2012.

[73] DA Fish, AM Brinicombe, ER Pike, and JG Walker, "Blind deconvolution by means of the richardson–lucy algorithm," *Journal of the Optical Society of America (A)*, vol. 12, no. 1, pp. 58–65, 1995.

[74] Dilip Krishnan and Rob Fergus, "Fast image deconvolution using hyper-Laplacian priors," in *NeurIPS*, 2009.

[75] Antoine Mousnier, Elif Vural, and Christine Guillemot, "Partial light field tomographic reconstruction from a fixed-camera focal stack," *arXiv preprint arXiv:1503.01903*, 2015.

[76] Nianyi Li, Jinwei Ye, Yu Ji, Haibin Ling, and Jingyi Yu, "Saliency detection on light field," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 2806–2813.

[77] Artemy Baxansky, "Apparatus, method, and manufacture for iterative auto-focus using depth-from-defocus," 2012, US Patent 8,218,061.

[78] Wei Zhang and Wai-Kuen Cham, "Single-image refocusing and defocusing," *IEEE Trans. on Image Processing*, vol. 21, no. 2, pp. 873–882, 2012.

[79] Yang Cao, Shuai Fang, and Zengfu Wang, "Digital multi-focusing from a single photograph taken with an uncalibrated conventional camera," *IEEE Trans. on image processing*, vol. 22, no. 9, pp. 3703–3714, 2013.

[80] Huixuan Tang, Scott Cohen, Brian Price, Stephen Schiller, and Kiriakos N Kutulakos, "Depth from defocus in the wild," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[81] Emma Alexander, Qi Guo, Sanjeev Koppal, Steven Gortler, and Todd Zickler, "Focal flow: Measuring distance and velocity with defocus and differential motion," in *EECV*, 2016, pp. 667–682.

[82] Supasorn Suwajanakorn, Carlos Hernandez, and Steven M Seitz, "Depth from focus with your mobile phone," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3497–3506.

[83] Marc Levoy, "Light fields and computational imaging," *Computer*, vol. 39, no. 8, pp. 46–55, 2006.

[84] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan, "Light field photography with a hand-held plenoptic camera," *Computer Science Technical Report CSTR*, vol. 2, no. 11, pp. 1–11, 2005.

[85] Caner Hazirbas, Sebastian Georg Soyer, Maximilian Christian Staab, Laura Leal-Taixé, and Daniel Cremers, "Deep depth from focus," in *ACCV*, 2018.

[86] Pratul P Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng, "Learning to synthesize a 4D RGBD light field from a single image," in *International Conference on Computer Vision (ICCV)*, 2017.

[87] Vivek Boominathan, Kaushik Mitra, and Ashok Veeraraghavan, "Improving resolution and depth-of-field of light field cameras using a hybrid imaging system," in *IEEE International Conference on Computational Photography (ICCP)*, 2014.

[88] Mark Sheinin, Yoav Y Schechner, and Kiriakos N Kutulakos, "Computational imaging on the electric grid," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[89] Hsing-Cheng Yu, Tzung-Yuan Lee, Shir-Kuan Lin, Li-Te Kuo, Shyh-Jier Wang, Jau-Jiu Ju, and Der-Ray Huang, "Low power consumption focusing actuator for a mini video camera," *Journal of Applied Physics*, vol. 99, no. 8, pp. 08R901, 2006.

[90] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[91] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., pp. 8024–8035. Curran Associates, Inc., 2019.

[92] Matthew Rhudy, Brian Bucci, Jeffrey Vipperman, Jeffrey Allanach, and Bruce Abraham, "Microphone array analysis methods using cross-correlations," in *ASME 2009 International Mechanical Engineering Congress and Exposition*. American Society of Mechanical Engineers, 2009, pp. 281–288.

[93] Google, "Google research: Android app to capture dual-pixel data," `https://github.com/google-research/google-research/tree/master/dual_pixels`, 2019, Last accessed: March, 2020.

[94] Tobias Plotz and Stefan Roth, "Benchmarking denoising algorithms with real photographs," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[95] Xiaojiao Mao, Chunhua Shen, and Yu-Bin Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *NeurIPS*, 2016.

[96] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.

[97] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[98] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber, "Training very deep networks," in *NeurIPS*, 2015.

[99] Augustus Odena, Vincent Dumoulin, and Chris Olah, "Deconvolution and checkerboard artifacts," *Distill*, vol. 1, no. 10, pp. e3, 2016.

[100] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *International Conference on Computer Vision (ICCV)*, 2015.

[101] François Chollet et al., "Keras," `https://keras.io`, 2015.

[102] Martin Abadi et. al., "TensorFlow: Large-scale machine learning on heterogeneous systems," https://www.tensorflow.org/, 2015.

[103] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al., "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing (TIP)*, vol. 13, no. 4, pp. 600–612, 2004.

[104] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[105] Anat Levin, Rob Fergus, Frédo Durand, and William T Freeman, "Image and depth from a conventional camera with a coded aperture," *ACM Transactions on sGraphics*, vol. 26, no. 3, pp. 70, 2007.

[106] Qing Guo, Wei Feng, Zhihao Chen, Ruijun Gao, Liang Wan, and Song Wang, "Effects of blur and deblurring to visual object tracking," *arXiv preprint arXiv:1908.07904*, 2019.

[107] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia, "Pyramid scene parsing network," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[108] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[109] Abdullah Abuolaim, Radu Timofte, Michael S Brown, et al., "NTIRE 2021 challenge for defocus deblurring using dual-pixel images: Methods and results," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021.

[110] Abdullah Abuolaim, Mauricio Delbracio, Damien Kelly, Michael S Brown, and Peyman Milanfar, "Learning to reduce defocus blur by realistically modeling dual-pixel data," in *International Conference on Computer Vision (ICCV)*, 2021.

[111] Maxim Maximov, Kevin Galim, and Laura Leal-Taixé, "Focus on defocus: bridging the synthetic to real domain gap for depth estimation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[112] Michael Potmesil and Indranil Chakravarty, "A lens and aperture camera model for synthetic image generation," *SIGGRAPH*, vol. 15, no. 3, pp. 297–305, 1981.

[113] Stephen Butterworth et al., "On the theory of filter amplifiers," *Wireless Engineer*, vol. 7, no. 6, pp. 536–541, 1930.

[114] Faisal Bukhari and Matthew N Dailey, "Automatic radial distortion estimation from a single image," *Journal of Mathematical Imaging and Vision*, vol. 45, no. 1, pp. 31–45, 2013.

[115] Andrew W Fitzgibbon, "Simultaneous linear estimation of multiple view geometry and lens distortion," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[116] Richard Hartley and Sing Bing Kang, "Parameter-free radial distortion correction with center of distortion estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 29, no. 8, pp. 1309–1321, 2007.

[117] B Prescott and GF McLean, "Line-based correction of radial lens distortion," *Graphical Models and Image Processing*, vol. 59, no. 1, pp. 39–47, 1997.

[118] 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)*, vol. 17, no. 10, pp. 1737–1754, 2008.

[119] 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)*, vol. 30, no. 2, pp. 299–314, 2007.

[120] Daniel Hernandez-Juarez, Lukas Schneider, Antonio Espinosa, David Vazquez, Antonio M. Lopez, Uwe Franke, Marc Pollefeys, and Juan Carlos Moure, "Slanted stixels: Representing san francisco's steepest streets," in *BMVC*, 2017.

[121] Wenguan Wang, Jianbing Shen, Fang Guo, Ming-Ming Cheng, and Ali Borji, "Revisiting video saliency: A large-scale benchmark and a new model," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[122] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2015.

[123] Zhengyang Lu and Ying Chen, "Single image super resolution based on a modified u-net with mixed gradient loss," *arXiv preprint arXiv:1911.09428*, 2019.

[124] Xiaoyang Guo, Hongsheng Li, Shuai Yi, Jimmy Ren, and Xiaogang Wang, "Learning monocular depth by distilling cross-domain stereo networks," in *European Conference on Computer Vision (ECCV)*, 2018.

[125] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb, "Learning from simulated and unsupervised images through adversarial training," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[126] Yinda Zhang, Neal Wadhwa, Sergio Orts-Escolano, Christian Häne, Sean Fanello, and Rahul Garg, "Du2net: Learning depth estimation from dual-cameras and dual-pixels," in *European Conference on Computer Vision (ECCV)*, 2020.

[127] Abdullah Abuolaim, Mahmoud Afifi, and Michael S Brown, "Improving single-image defocus deblurring: How dual-pixel images help through multi-task learning," in *Winter Conference on Applications of Computer Vision (WACV)*, 2022.

[128] Ross Girshick, "Fast R-CNN," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[129] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[130] Shikun Liu, Edward Johns, and Andrew J Davison, "End-to-end multi-task learning with attention," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[131] Xuaner Zhang, Ren Ng, and Qifeng Chen, "Single image reflection separation with perceptual losses," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[132] Yu Li and Michael S Brown, "Exploiting reflection change for automatic reflection removal," in *International Conference on Computer Vision (ICCV)*, 2013.

[133] Jie Yang, Dong Gong, Lingqiao Liu, and Qinfeng Shi, "Seeing deeply and bidirectionally: A deep learning approach for single image reflection removal," in *European Conference on Computer Vision (ECCV)*, 2018.

[134] Xiaojie Guo, Xiaochun Cao, and Yi Ma, "Robust separation of reflection from multiple images," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[135] Abdullah Abuolaim, Mahmoud Afifi, and Michael S Brown, "Multi-view motion synthesis via applying rotated dual-pixel blur kernels," in *Winter Conference on Applications of Computer Vision Workshops (WACVW)*, 2022.

[136] Carlos Hernández, "Lens blur in the new google camera app," `http://research.googleblog.com/2014/04/lens-blur-in-new-google-camera-app.html`, 2014.

[137] Hyowon Ha, Sunghoon Im, Jaesik Park, Hae-Gon Jeon, and In So Kweon, "High-quality depth from uncalibrated small motion clip," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[138] Fisher Yu and David Gallup, "3d reconstruction from accidental motion," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[139] Andrey Ignatov, Jagruti Patel, and Radu Timofte, "Rendering natural camera bokeh effect with deep learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020.

[140] Ravi Garg, BG Vijay Kumar, Gustavo Carneiro, and Ian Reid, "Unsupervised CNN for single view depth estimation: Geometry to the rescue," in *European Conference on Computer Vision*, 2016.

[141] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[142] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe, "Unsupervised learning of depth and ego-motion from video," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[143] Junyuan Xie, Ross Girshick, and Ali Farhadi, "Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks," in *European Conference on Computer Vision (ECCV)*, 2016.

[144] Jonathan T Barron and Jitendra Malik, "Shape, illumination, and reflectance from shading," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 37, no. 8, pp. 1670–1687, 2014.

[145] Berthold KP Horn, "Obtaining shape from shading information," *The psychology of computer vision*, pp. 115–155, 1975.

[146] David Eigen, Christian Puhrsch, and Rob Fergus, "Depth map prediction from a single image using a multi-scale deep network," *arXiv preprint arXiv:1406.2283*, 2014.

[147] Derek Hoiem, Alexei A Efros, and Martial Hebert, "Automatic photo pop-up," in *SIGGRAPH*. 2005.

[148] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid, "Learning depth from single monocular images using deep convolutional neural fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 38, no. 10, pp. 2024–2039, 2015.

[149] Ashutosh Saxena, Min Sun, and Andrew Y Ng, "Learning 3-d scene structure from a single still image," in *International Conference on Computer Vision (ICCV)*, 2007.

[150] Xiaoyong Shen, Aaron Hertzmann, Jiaya Jia, Sylvain Paris, Brian Price, Eli Shechtman, and Ian Sachs, "Automatic portrait segmentation for image stylization," in *Computer Graphics Forum*. Wiley Online Library, 2016, vol. 35, pp. 93–102.

[151] Xiaoyong Shen, Xin Tao, Hongyun Gao, Chao Zhou, and Jiaya Jia, "Deep automatic portrait matting," in *European Conference on Computer Vision (ECCV)*, 2016.

[152] Kyle Krafka, Aditya Khosla, Petr Kellnhofer, Harini Kannan, Suchendra Bhandarkar, Wojciech Matusik, and Antonio Torralba, "Eye tracking for everyone," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[153] Codruta O Ancuti, Cosmin Ancuti, Florin-Alexandru Vasluianu, Radu Timofte, et al., "NTIRE 2021 nonhomogeneous dehazing challenge report," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021.

[154] Majed El Helou, Ruofan Zhou, Sabine Süsstrunk, Radu Timofte, et al., "NTIRE 2021 depth guided image relighting challenge," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021.

[155] Seungjun Nah, Sanghyun Son, Suyoung Lee, Radu Timofte, Kyoung Mu Lee, et al., "NTIRE 2021 challenge on image deblurring," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021.

[156] Jerrick Liu, Oliver Nina, Radu Timofte, et al., "NTIRE 2021 multi-modal aerial view object classification challenge," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021.

[157] Andreas Lugmayr, Martin Danelljan, Radu Timofte, et al., "NTIRE 2021 learning the super-resolution space challenge," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021.

[158] Ren Yang, Radu Timofte, et al., "NTIRE 2021 challenge on quality enhancement of compressed video: Methods and results," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021.

[159] Sanghyun Son, Suyoung Lee, Seungjun Nah, Radu Timofte, Kyoung Mu Lee, et al., "NTIRE 2021 challenge on video super-resolution," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021.

[160] Jinjin Gu, Haoming Cai, Chao Dong, Jimmy S. Ren, Yu Qiao, Shuhang Gu, Radu Timofte, et al., "NTIRE 2021 challenge on perceptual image quality assessment," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021.

[161] Goutam Bhat, Martin Danelljan, Radu Timofte, et al., "NTIRE 2021 challenge on burst super-resolution: Methods and results," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021.

[162] Eduardo Pérez-Pellitero, Sibi Catley-Chandar, Aleš Leonardis, Radu Timofte, et al., "NTIRE 2021 challenge on high dynamic range imaging: Dataset, methods and results," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2021.

[163] Zhou Wang, Eero P Simoncelli, and Alan C Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003.* IEEE, 2003, vol. 2, pp. 1398–1402.

[164] Radu Timofte, Rasmus Rothe, and Luc Van Gool, "Seven ways to improve example-based single image super resolution," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[165] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[166] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017.

[167] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, 2015, vol. 2.

[168] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao, "Multi-stage progressive image restoration," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[169] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu, "Image super-resolution using very deep residual channel attention networks," in *European Conference on Computer Vision (ECCV)*, 2018.

[170] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu, "Dual attention network for scene segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[171] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon, "Cbam: Convolutional block attention module," in *European Conference on Computer Vision (ECCV)*, 2018.

[172] Stephan Brehm, Sebastian Scherer, and Rainer Lienhart, "High-resolution dual-stage multi-level feature aggregation for single image and video deblurring," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020.

[173] Abhijith Punnappurath and Michael S. Brown, "Reflection removal using a dual-pixel sensor," in *Computer Vision and Pattern Recognition*, 2019.

[174] Daniel Scharstein and Richard Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1–3, pp. 7–42, 2002.

[175] Richard Hartley and Andrew Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2003.

[176] B. K.P. Horn, "Shape from shading: A method for obtaining the shape of a smooth opaque object from one view," Tech. Rep., 1970, Artificial Intelligence Laboratory, Massachusetts Institute of Technology (MIT).

[177] Michael Brady and Alan Yuille, "An extremum principle for shape from contour," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 288–301, 1983.

[178] Ruzena Bajcsy and Lawrence Lieberman, "Texture gradient as a depth cue," *Computer Graphics and Image Processing*, vol. 5, no. 1, pp. 52 – 67, 1976.

[179] Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng, "Learning depth from single monocular images," in *Neural Information Processing Systems*, 2005.

[180] A. Saxena, M. Sun, and A. Y. Ng, "Make3D: Learning 3D scene structure from a single still image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 824–840, 2009.

[181] L'ubor Ladický, Jianbo Shi, and Marc Pollefeys, "Pulling things out of perspective," in *Computer Vision and Pattern Recognition*, 2014.

[182] Xiu Li, Hongwei Qin, Yangang Wang, Yongbing Zhang, and Qionghai Dai, "DEPT: Depth estimation by parameter transfer for single still images," in *Asian Conference on Computer Vision*, 2015.

[183] S. Choi, D. Min, B. Ham, Y. Kim, C. Oh, and K. Sohn, "Depth analogy: Data-driven approach for single image depth estimation using gradient samples," *IEEE Transactions on Image Processing (TIP)*, vol. 24, no. 12, pp. 5953–5966, 2015.

[184] J. Konrad, M. Wang, P. Ishwar, C. Wu, and D. Mukherjee, "Learning-based, automatic 2D-to-3D image and video conversion," *IEEE Transactions on Image Processing (TIP)*, vol. 22, no. 9, pp. 3485–3496, 2013.

[185] Jianping Shi, Xin Tao, Li Xu, and Jiaya Jia, "Break ames room illusion: Depth from general single images," *ACM Transactions on Graphics*, vol. 34, no. 6, pp. 225:1–225:11, 2015.

[186] R. Ranftl, V. Vineet, Q. Chen, and V. Koltun, "Dense monocular depth estimation in complex dynamic scenes," in *Computer Vision and Pattern Recognition*, 2016.

[187] Christian Häne, L'ubor Ladicky, and Marc Pollefeys, "Direction matters: Depth estimation with a surface normal classifier," in *Computer Vision and Pattern Recognition*, 2015.

[188] David Eigen, Christian Puhrsch, and Rob Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Neural Information Processing Systems*, 2014.

[189] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao, "Deep ordinal regression network for monocular depth estimation," in *Computer Vision and Pattern Recognition*, 2018.

[190] J. Li, R. Klein, and A. Yao, "A two-streamed network for estimating fine-scaled depth maps from single rgb images," in *International Conference on Computer Vision*, 2017, pp. 3392–3400.

[191] A. Roy and S. Todorovic, "Monocular depth estimation using neural regression forest," in *Computer Vision and Pattern Recognition*, 2016.

[192] Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid, "Learning depth from single monocular images using deep convolutional neural fields," *IEEE Transactions on Pattern Analysis and Machince Intelligence*, vol. 38, no. 10, pp. 2024–2039, 2016.

[193] A. Atapour-Abarghouei and T.P. Breckon, "Real-time monocular depth estimation using synthetic data with domain adaptation," in *Computer Vision and Pattern Recognition*, 2018.

[194] Yuliang Zou, Zelun Luo, and Jia-Bin Huang, "DF-Net: Unsupervised joint learning of depth and flow using cross-task consistency," in *European Conference on Computer Vision*, 2018.

[195] Nikolaus Mayer, Eddy Ilg, Philipp Fischer, Caner Hazirbas, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox, "What makes good synthetic training data for learning disparity and optical flow estimation?," *International Journal on Computer Vision*, vol. 126, no. 9, pp. 942–960, 2018.

[196] J. N. Kundu, P. K. Uppala, A. Pahuja, and R. V. Babu, "AdaDepth: Unsupervised content congruent adaptation for depth estimation," in *Computer Vision and Pattern Recognition*, 2018.

[197] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow, "Digging into self-supervised monocular depth prediction," in *International Conference on Computer Vision*, 2019.

[198] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Computer Vision and Pattern Recognition*, 2017.

[199] Reza Mahjourian, Martin Wicke, and Anelia Angelova, "Unsupervised learning of depth and ego-motion from monocular video using 3D geometric constraints," in *Computer Vision and Pattern Recognition*, 2018.

[200] Huaizu Jiang, Gustav Larsson, Michael Maire, Greg Shakhnarovich, and Erik Learned-Miller, "Self-supervised relative depth learning for urban scene understanding," in *European Conference on Computer Vision*, 2018.

[201] P. Grossmann, "Depth from focus," *Pattern Recognition Letters*, vol. 5, no. 1, pp. 63 – 69, 1987.

[202] Yoav Y. Schechner and Nahum Kiryati, "Depth from defocus vs. stereo: How different really are they?," *International Journal of Computer Vision*, vol. 39, pp. 141–162, 2000.

[203] A. N. Rajagopalan, S. Chaudhuri, and U. Mudenagudi, "Depth estimation and image restoration using defocused stereo pairs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1521–1525, 2004.

[204] Arnav V. Bhavsar and A. N. Rajagopalan, "Towards unrestrained depth inference with coherent occlusion filling," *International Journal of Computer Vision*, vol. 97, pp. 167–190, 2011.

[205] Feng Li, Jian Sun, Jue Wang, and Jingyi Yu, "Dual-focus stereo imaging," *Journal of Electronic Imaging*, vol. 19, 2010.

[206] C. Chen, H. Zhou, and T. Ahonen, "Blur-aware disparity estimation from defocus stereo images," in *International Conference on Computer Vision*, 2015.

[207] C. Paramanand and A. N. Rajagopalan, "Depth from motion and optical blur with an unscented Kalman filter," *IEEE Transactions on Image Processing (TIP)*, vol. 21, no. 5, pp. 2798–2811, 2012.

[208] H. Jeon, J. Park, G. Choe, J. Park, Y. Bok, Y. Tai, and I. S. Kweon, "Accurate depth map estimation from a lenslet light field camera," in *Computer Vision and Pattern Recognition*, 2015.

[209] H. Jeon, J. Park, G. Choe, J. Park, Y. Bok, Y. Tai, and I. S. Kweon, "Depth from a light field image with learning-based matching costs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 2, pp. 297–310, 2019.

[210] M. Landy and J. A. Movshon, *The Plenoptic Function and the Elements of Early Vision*, MITP, 1991.

[211] Fahim Mannan and Michael S Langer, "Blur calibration for depth from defocus," in *Conference on Computer and Robot Vision (CRV)*, 2016.

[212] Zhe Hu and Ming-Hsuan Yang, "Good regions to deblur," in *European Conference on Computer Vision*, 2012.

[213] Gopal Harikumar and Yoram Bresler, "Perfect blind restoration of images blurred by multiple filters: theory and efficient algorithms," *IEEE Transactions on Image Processing (TIP)*, vol. 8, no. 2, pp. 202–219, 1999.

[214] Tao Xian and Murali Subbarao, "Depth-from-defocus: Blur equalization technique," in *SPIE: Society of Photo-Optical Instrumentation Engineers*, 2006.

[215] Jonathan T. Barron and Ben Poole, "The fast bilateral solver," *European Conference on Computer Vision*, 2016.

[216] Kaiming He, Jian Sun, and Xiaoou Tang, "Guided image filtering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 6, pp. 1397–1409, 2013.

[217] Atsushi Ito, Salil Tambe, Kaushik Mitra, Aswin C Sankaranarayanan, and Ashok Veeraraghavan, "Compressive epsilon photography for post-capture control in digital imaging," *ACM Transactions on Graphics*, vol. 33, no. 4, pp. 88:1–88:12, 2014.

[218] Neel Joshi, Richard Szeliski, and David J Kriegman, "Psf estimation using sharp edge prediction," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[219] Mahmoud Afifi, Abdelrahman Abdelhamed, Abdullah Abuolaim, Abhijith Punnappurath, and Michael S Brown, "Cie xyz net: Unprocessing images for low-level computer vision tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[220] Hakki Can Karaimer and Michael S Brown, "A software platform for manipulating the camera imaging pipeline," in *European Conference on Computer Vision (ECCV)*, 2016.

[221] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand, "Learning photographic global tonal adjustment with a database of input / output image pairs," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[222] Rang MH Nguyen and Michael S Brown, "Raw image reconstruction using a self-contained sRGB-JPEG image with only 64 KB overhead," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[223] Tim Brooks, Ben Mildenhall, Tianfan Xue, Jiawen Chen, Dillon Sharlet, and Jonathan T Barron, "Unprocessing images for learned raw denoising," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[224] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao, "CycleISP: Real image restoration via improved data synthesis," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[225] R. M. H. Nguyen, D. K. Prasad, and M. S. Brown, "Raw-to-raw: Mapping between image sensor color responses," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[226] 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.

[227] Seonghyeon Nam and Seon Joo Kim, "Modelling the scene dependent imaging in cameras with a deep neural network," in *International Conference on Computer Vision (ICCV)*, 2017.

[228] Yuanming Hu, Baoyuan Wang, and Stephen Lin, "FC4: Fully convolutional color constancy with confidence-weighted pooling," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[229] Mahmoud Afifi and Michael S Brown, "Sensor-independent illumination estimation for dnn models," *arXiv preprint arXiv:1912.06888*, 2019.

[230] Seon Joo Kim, Hai Ting Lin, Zheng Lu, Sabine Süsstrunk, Stephen Lin, and Michael S Brown, "A new in-camera imaging model for color computer vision and its application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2289–2302, 2012.

[231] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown, "A high-quality denoising dataset for smartphone cameras," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[232] 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 Electronic Devices*, vol. 54, no. 6, pp. 1321–1329, 2007.

[233] Samuel W Hasinoff, Frédo Durand, and William T Freeman, "Noise-optimal capture for high dynamic range photography," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 553–560.

[234] Haiting Lin, Seon Joo Kim, S. Süsstrunk, and M. S. Brown, "Revisiting radiometric calibration for color computer vision," in *International Conference on Computer Vision (ICCV)*, 2011.

[235] A. Chakrabarti, Ying Xiong, Baochen Sun, T. Darrell, D. Scharstein, T. Zickler, and K. Saenko, "Modeling radiometric uncertainty for vision with tone-mapped color images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2185–2198, 2014.

[236] 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)*, vol. 35, no. 6, pp. 192:1–192:12, 2016.

[237] Mahmoud Afifi, Brian Price, Scott Cohen, and Michael S Brown, "When color constancy goes wrong: Correcting improperly white-balanced images," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[238] Matthew Anderson, Ricardo Motta, Srinivasan Chandrasekar, and Michael Stokes, "Proposal for a standard default color space for the internet - srgb," in *Color and Imaging Conference*, 1996, pp. 238–245.

[239] Marc Ebner, *Color Constancy*, vol. 6, John Wiley & Sons, 2007.