UNIVERSITY OF OULU

FACULTY OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Ahmed Alhawwary

DEPTH AND IMU AIDED IMAGE DEBLURRING BASED ON DEEP LEARNING

Master's Thesis
Degree Programme in Computer Science and Engineering
June 2020

Alhawwary A. (2020) Depth and IMU Aided Image Deblurring Based On Deep Learning. University of Oulu, Degree Programme in Computer Science and Engineering, 65 p.

ABSTRACT

With the wide usage and spread of camera phones, it becomes necessary to tackle the problem of the image blur. Embedding a camera in those small devices implies obviously small sensor size compared to sensors in professional cameras such as full-frame Digital Single-Lens Reflex (DSLR) cameras. As a result, this can dramatically affect the collected amount of photons on the image sensor. To overcome this, a long exposure time is needed, but with slight motions that often happen in handheld devices, experiencing image blur is inevitable. Our interest in this thesis is the motion blur that can be caused by the camera motion, scene (objects in the scene) motion, or generally the relative motion between the camera and scene. We use deep neural network (DNN) models in contrary to conventional (non DNN-based) methods which are computationally expensive and time-consuming. The process of deblurring an image is guided by utilizing the scene depth and camera's inertial measurement unit (IMU) records. One of the challenges of adopting DNN solutions is that a relatively huge amount of data is needed to train the neural network. Moreover, several hyperparameters need to be tuned including the network architecture itself.

To train our network, a novel and promising method of synthesizing spatially-variant motion blur is proposed that considers the depth variations in the scene, which showed improvement of results against other methods. In addition to the synthetic dataset generation algorithm, a real blurry and sharp dataset collection setup is designed. This setup can provide thousands of real blurry and sharp images which can be of paramount benefit in DNN training or fine-tuning.

Keywords: motion blur, deep neural network, synthetic dataset, data capturing

TABLE OF CONTENTS

FOREWORD

This master's thesis was written at the Center for Machine Vision and Signal Analysis (CMVS) at the University of Oulu. First of all, I would like to express my gratitude to Professor Janne Heikkilä and his PhD student Janne Mustaniemi for their invaluable supervision, guidance, and support. I greatly benefited from their knowledge. It has been a pleasure to work in the research group. My sincere thank goes to Dr Matteo Pedone for reviewing my work. Also, I would like to thank our industrial project partners for their insights and discussions. Finally, I would like to thank my lovely princess Nada Ebaid, my family and friends for their continuous encouragement and support.

Oulu, June 18th, 2020

Ahmed Alhawwary

LIST OF ABBREVIATIONS AND SYMBOLS

DNN	Deep Neural Network
DL	Deep Learning
DSLR	Digital Single-Lens Reflex
CNN	Convolutional Neural Networks
IMU	Inertial Measurement Unit
DOF	Depth of Field
i.i.d.	Independent and Identically Distributed
MAP	Maximum A Posteriori
SD	Standard Deviation
RNN	Recurrent Neural Networks
GAN	Generative Adversarial Network
MSE	Mean Squared Error

K	camera intrinsic matrix
f	focal length
N	number of samples
t	time
τ	exposure time
σ^2	variance
∂	partial derivative operator
$\int$	integration operator
$\odot$	element-wise product operator

1. INTRODUCTION

Handheld and mobile cameras become widely spread nowadays, and it is hard to find a modern mobile phone that does not contain at least one camera. Embedding a camera in devices that have a crowd of other complicated parts and the limit of the mobile size, the camera sensors are very small compared to the full-frame sensors in the professional cameras such as DSLR cameras. This imposes challenges since a lower amount of photons will be collected by those sensors. However, the exposure time can be increased to allow more photons to fall onto the sensors. Unfortunately, this may not be possible without experiencing image blur from slight camera motion or object motion especially at low light conditions. There are several types of image blur, and some of them are created on purpose to result in aesthetic work, but our interest here is the motion blur. Motion blur can result from the camera motion while the scene is static, the scene motion while the camera is steady, or both of them, which cause very complicated and spatially-variant blur. Moreover, when we have a relative camera-scene translation, and significant variation in the scene depth with very close and far objects, the depth becomes vital in determining the blur variations across the scene.

The problem of image blur has been widely studied through the literature and several methods and techniques were proposed to remove or relieve the image blur and enhance the image quality. With the success of the DNN models in a broad domain of computer vision tasks, deblurring the image with several deep learning techniques becomes attempting and showed promising results in comparison to conventional (non deep learning-based) methods. This motivates us to undertake further exploration of the exiting deep learning architectures and models and how to feed the additional information to the learning models. The conventional methods are computationally expensive and time-consuming compared to the deep learning-based techniques. On the other hand, deep learning-based methods require a relatively huge amount of diverse data samples to be effective for network training. Moreover, several hyperparameters need to be tuned such as learning rate, number of layers, etc.

In this thesis, we propose a deep learning-based solution for deblurring images suffering from spatially-variant blur with the help of the scene depth and camera's IMU measurements. Modern mobiles are usually equipped by IMU units which can be utilized to determine the rotations and translations of the camera. Moreover, the new mobiles have been increasingly equipped with multiple cameras or a depth sensor. However, the range of the depth that can be retrieved by those multiple cameras are limited due to the small baseline between the cameras.

Moreover, a novel technique is proposed for synthesizing spatially-variant blurred images by utilizing the depth map and sampled IMU measurements. Our experiments showed improvements in results of deblurring real-world blurry images compared to other deep learning-based techniques.

Besides the synthetic dataset generation technique, a multi-camera setup is proposed to collect real blurry and sharp image pairs and their corresponding depth maps and IMU measurements. While collecting such data is extremely challenging, we demonstrate a wide step of development in that direction. Moreover, we spot the lights on the challenges that hinder the process of collecting those images. Such dataset can be utilized for finetuning the pre-trained model to improve the results of deblurring real-world blurry images.

2. IMAGE BLUR

Blur, a word that usually relates to image capturing process, is a type of degradation in image. Blur in images can be seen as annoying thing and degradation in image quality, while in other cases, it is seen as plausible thing and aesthetic work. But what is blur? The best way to describe image blur is to list its types and explore the common things among those types.

2.1. Blur Types

Image blur can be divided into 4 types based on their cause. They are motion blur, defocus blur, atmospheric blur, intrinsic physical blur. In [1], they additionally divided the first type, motion blur, to two more types: object motion blur, and camera shake blur. Since our interest in this thesis is the motion blur, we discuss it in depth separately in the next section, and talk here briefly about the other types.

- Defocus blur occurs when the camera is either able to bring the foreground or background only into focus, but not both, as shown in Figure 1. While this can be sometimes desirable and considered as artistic feature in the image, it is considered as defect in other images where the details of both background and foreground are needed. The defocus happens due to the shallow depth of field (DOF) of the camera, which is mainly determined by the lens aperture size. Defocus blur increases when the aperture size increases.

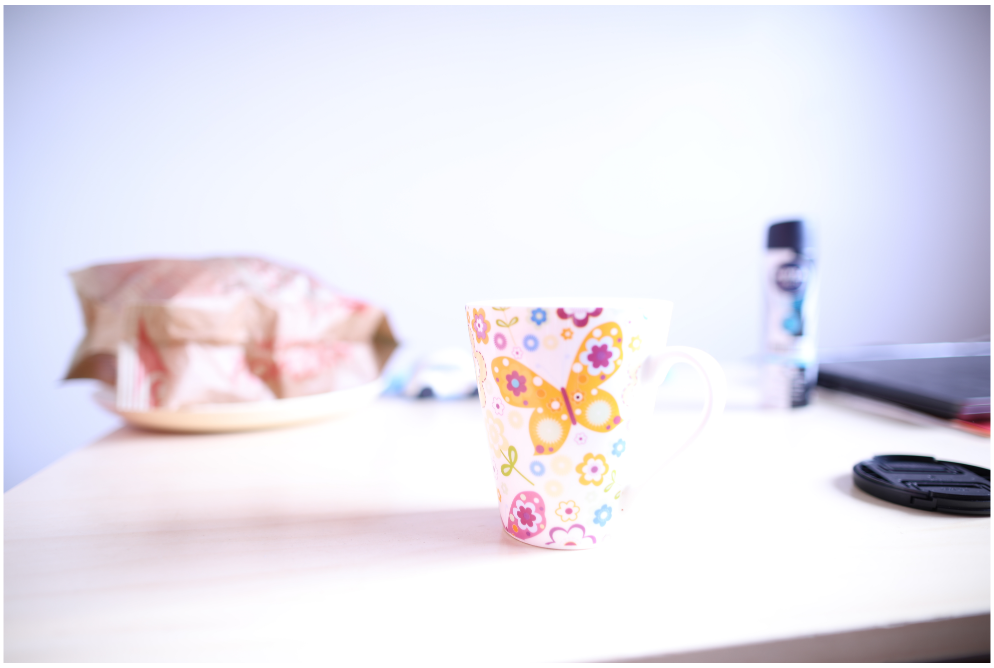


Figure 1. Out-of-focus blur. The objects in the background are out of focus.

- Atmospheric blur is usually observed when the captured scene includes very far objects. Generally, it happens because the atmospheric particulate matters hinder the photons causing variation in the refractive index along the path to the camera sensor. An example of such type of blur is illustrated in Figure 2.

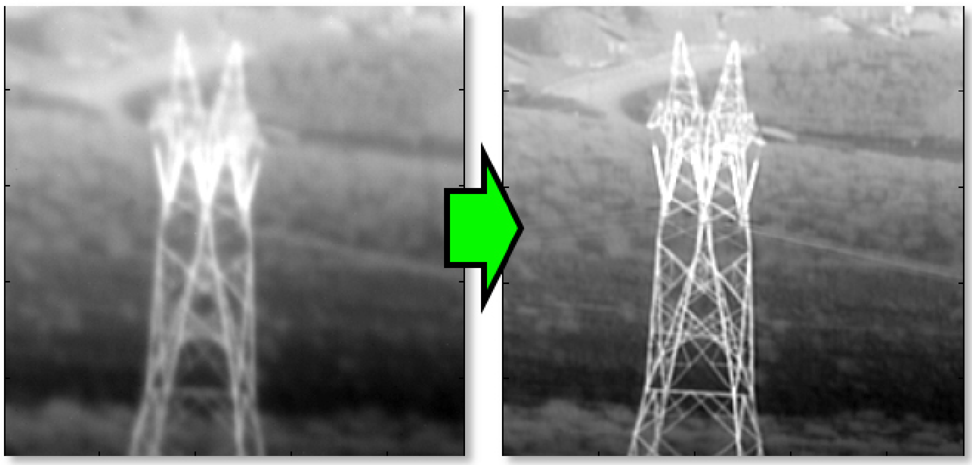


Figure 2. Atmospheric blur on the left, restored image on the right [2].

- Intrinsic physical blur happens due to inherent issues in the imaging system itself. For instance, the chromatic aberration, which is one type of lens (optical) aberration, results from the varying refractive index of the lens for the incident white light, making it hard for the light to converge into one point on the focal plane. Other sources can be light diffraction characteristics, sensor resolution and anti-aliasing filter (digital imaging). An example of optical aberration is shown in Figure 3.

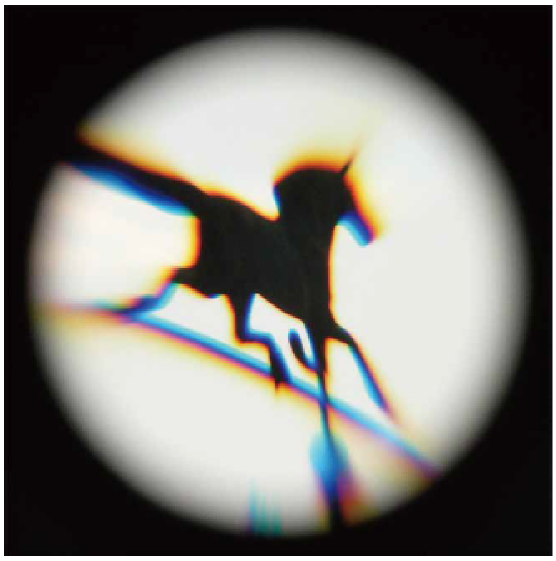


Figure 3. Optical aberration [1].

2.2. Motion Blur

2.2.1. Why It Occurs?

Motion blur might happen when the camera moves while the scene is static, or when the camera is steady but the scene is dynamic, or both of them are moving. Generally, this can be called camera-scene motion, where there is relative motion between the camera and the scene. When we refer to scene motion here, we mean objects that belong to that scene.

During the exposure time, and supposing camera-scene motion, then the reflected photons from points in the scene fall into different positions of the camera sensor. This can be prevented by decreasing the exposure time such that it is faster than the motion. However, with situations of low light conditions, this may be not possible, since longer exposure time is required for the sensor to collect enough photons from the scene. Moreover, in the modern phone cameras, the imaging system is embedded in a very limited space with a crowd of other electronic components, which implies that the sensor is significantly smaller than the full-frame camera sensor such as DSLR camera. This will dramatically affect the amount of collected photons in the same exposure time duration of the full-frame camera. Considering those aspects, it may not be applicable to make the exposure time faster, and it should be long enough to collect enough photons for the capture. Therefore with the handheld (or mobile phone) camera motion and bad lighting conditions, avoiding the image blur might be inevitable.

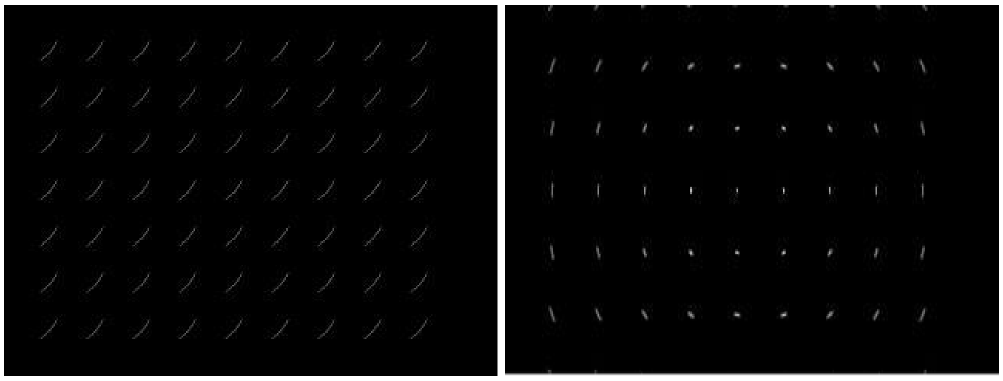
The blur kernel or point spread function (PSF) is a function that describes the blur across the image and how each point (pixel) in the image moves (spreads). The blur across the scene can be uniform (spatially-invariant) or non-uniform (spatially-variant). Uniform blur means that each point in the scene has the same blur kernel, while they have varying blur kernels in the non-uniform case.

One simple example of uniform blur is as following: suppose we have a plane (e.g. a paper) which is parallel to the image plane of the camera and we translate in a direction parallel to the image plane, then the blur is uniform across all the points in the scene, as shown in Figure 4a. The previous example is a very special and restricted case. In real world, the blur is often non-uniform. One intuitive example of non-uniform blur is the in-plane (in terms of image plane) rotation of the camera (i.e. around the camera z-axis) where blur increases as we go away from the image center, as shown in Figure 4b.

2.2.2. Blur Kernel Determinants

There are several factors that affect the shape of the blur kernel through the motion blurry image. First, we discuss the camera shake only situation, and then move to the scene motion situation.

As we discussed in the previous subsection, the shake blur comes form changing camera pose during the exposure time. Camera pose is determined by the orientation and position of its 3D frame relative to a 3D world reference frame.



(a) Uniform blur (b) Non-uniform blur

Figure 4. Uniform blur vs non-uniform blur. It is useful to observe the motion path of a point in the scene by light streaks. Imagine that the white points are light points that are co-planar and parallel to the image plane. (a) shows a uniform blur resulting from in-plane camera translation. (b) shows an non-uniform blur resulting from in-plane camera.

Motion trajectory and velocity. Intuitively, the velocity of motion and the path camera takes during specific exposure time are factors that determine the blur kernel shape.

Scene depth. When camera undergoes pure rotation around its optical center, it reveals no additional information about the scene depth. For instance, when the camera rotates around its Y-axis, it is observed that the occlusions do not change and no additional information is provided about the occluded parts, and naturally the boundaries only differ from a view to another. Moreover, the views are related to each other with a homography. This is also the case when scene points share a planar surface, which implies that there is no occlusion in the scene, but it does not matter in that case how the camera moves and the views are still related with a homography. On the other hand, when the camera undergoes translation and we have arbitrary (irregular) 3D scene with objects located in different depth, the occluded parts of the background objects change and reveal new information about the 3D scene. Motion parallax is considered as monocular cue for depth perception. The close object appears to move faster than the farther objects. From the definition, it is clear that motion parallax does not happen with pure camera rotation around its center, but with the camera translation. Based on this observation, and supposing that the camera is translating along its X-axis for instance, it is expected that the near objects undergo larger blur than the farther objects, as in Figure 5. This indicates how depth can cause blur variation across the scene, but does not mean that always the objects with significant depth will undergo lesser blur than the near objects, for instance, if the camera rotates around its Y-axis in one direction while translating in X-axis in the opposite direction. However this still means that the depth is in effect.

In [3], they show that the most of blur in handheld camera comes from small rotations around the camera center, and that the blur from translations requires significant translation in the device motion which usually does not happen while capturing images.

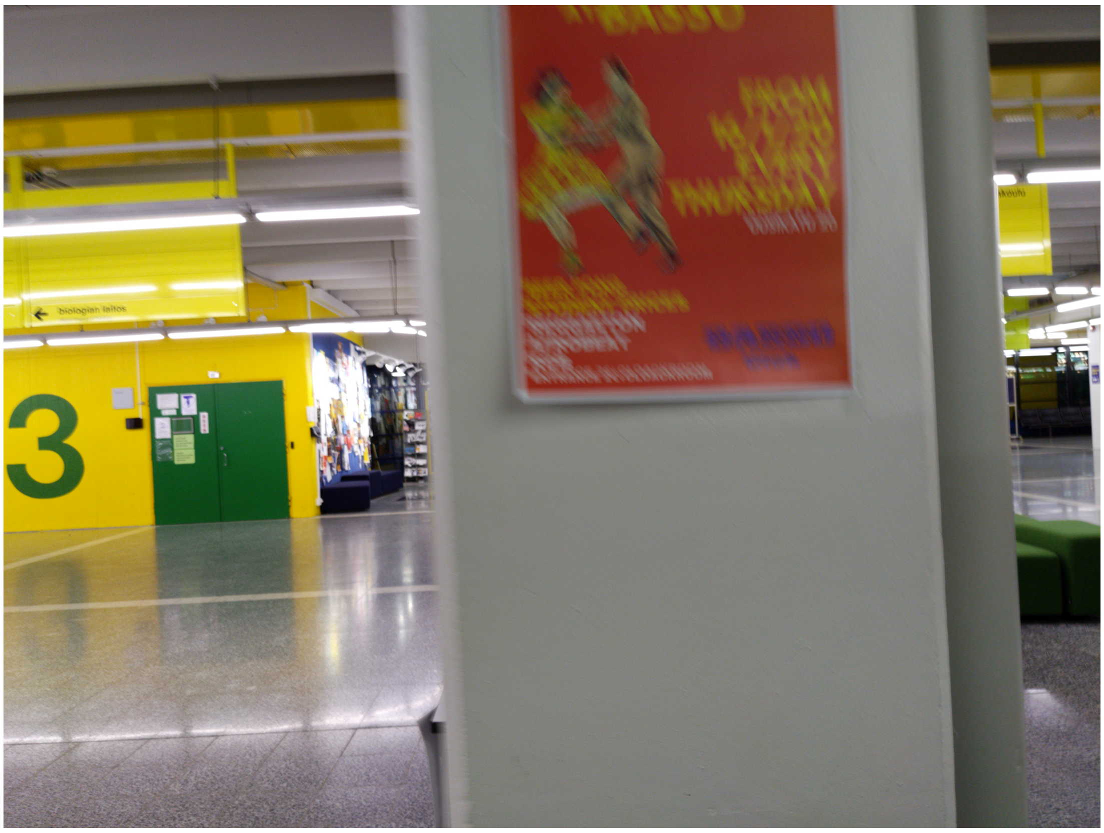


Figure 5. Image with depth-dependent blur. The foreground poster is significantly blurred, while the background is almost sharp.

Intrinsic matrix contains intrinsic parameters related to the camera, to distinguish it from the extrinsic parameters which is basically the camera pose (rotations and translations). In practice, usually intrinsic camera matrix K is defined as the following:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where f_x and f_y are the focal length in the x and y direction respectively, and c_x and c_y are the principal point coordinates in the x and y direction respectively.

Object Motion. When the situation includes object motion in addition to the camera motion, it becomes more complicated. The blur kernel shape over the moving object depends on the motion velocity (its quantity and direction), and object distance from the camera. For instance, if there are two objects moving with the same velocity but with different distance from the camera, then it is expected to experience more blur on the closer object to the camera (parallax motion).

The blur model is a mathematical model that describes the characteristics of blur. Such modelling is of vital role in understanding the behaviour of the blur and building effective solution for it. The blur model can be exploited in removing the blur from image, or the inverse direction, in synthesizing a type of blur in image on a purpose. Moreover, in the new trends of using learning-based algorithms and models especially

DNN models for image deblurring and image enhancement, it becomes essential to provide a huge number of training samples of sharp and blurry pairs for those models to be able to effectively remove blur from unseen images. Since it is extremely hard - but still possible - to acquire both sharp and blurry images for the same scene in the real world, it is important to find a way to synthesize a blur from the sharp image instances. We will start by describing the general blur model, and then discuss the motion blur modelling.

2.2.3. Spatially-Invariant Blur Model

Usually the general image blur is described as following:

$$B = I * S + n \tag{2}$$

where B is the blurred image, I is the latent sharp image, $*$ is the convolution operator, S is the blur kernel (or PSF) and n is a noise term, that is usually additive white Gaussian noise. However this model assumes only that blur in the whole scene is uniform and all pixels are blurred the same way. In other words, it assumes uniform blur kernel or spatially-invariant blur. Therefore it is not suitable for modelling motion blur which often introduces a non-uniform blur or spatially-variant blur.

2.2.4. Spatially-Variant Blur Model

A more comprehensive motion blur model assumes that the blur is an integration of the intermediate scene images along the camera motion path [4, 3, 5]:

$$B = \int_{t=0}^{\tau} I(\theta_t) dt + n \tag{3}$$

where $I(\theta_t)$ is the image instance when the camera has pose θ_t at time instant t. Since we deal with a digital world with finite capacity, time discretization is applied:

$$B = \sum_{j=0}^{N-1} \mu_j I(\theta_j) + n \tag{4}$$

where N is the sampled time instants, and μ_j is the weight equivalent to the portion of the time camera spends at pose θ_j during the exposure time.

Let p_0 be a projected point in homogeneous coordinates $(x_0, y_0, 1)^T$, and a result of projecting a 3D world point $P = (X, Y, Z, 1)^T$ at the beginning of exposure. The projection p' of the same 3D point as camera moves is related with p_0 through a planar homography matrix $H(\theta_j, d)$ at time instant t_j and depth d:

$$p(x, y, \theta_j) = H(\theta_j, d) * p(x, y, \theta_0) \tag{5}$$

where H is defined as the following:

$$H(\theta_j, d) = K^{-1}(R(\theta_j) + \frac{1}{d}T(\theta_j)r^T)K \tag{6}$$

where K is the camera intrinsic matrix. $R(\theta_j)$ and $T(\theta_j)$ are camera rotation and translation at pose θ_j respectively, and r is the scene normal and usually assumed to be a unit vector orthogonal to image plane for all points, r = $[0, 0, 1]^T$ (i.e. fronto-parallel scene).

From equation (6), it can be noticed that when we have constant depth or no translation, then all projected pixels at pose θ_j share the same homography matrix. However, when we have arbitrary depth values (not planar scene), then every pixel has its own homography matrix. Moreover, it is noticed that the scene depth variation affects only when we have camera translation.

3. CONVENTIONAL IMAGE DEBLURRING

3.1. Problem Definition

Image deblurring is the process of recovering a latent sharp image from a blurry one. If the blur kernel (or the PSF for every pixel) is known, the process is referred to as non-blind image deblurring. In contrast, blind image deblurring aims to recover the image without the existence of the blur kernel in the first place. Blind deblurring is an inherently ill-posed problem since it is under-constrained, and there are many combinations of blur kernel and latent image pairs that can result in the same blurry image. This can be clearly noted from equation (2). Figure 6 illustrates this ill-posedness. To constrain the solution space, several additional information and prior knowledge are used, such as natural image statistics [6, 7, 8, 9], unnatural L0 priors [10], dark channel priors [11], color line priors [12], scene depth maps [13, 14, 5], camera motion information [15, 16, 17, 18, 19], and other image observations e.g. low/high resolution image pairs [20, 21], noisy/blurry image pairs [22] , and blurred image sequence [23].

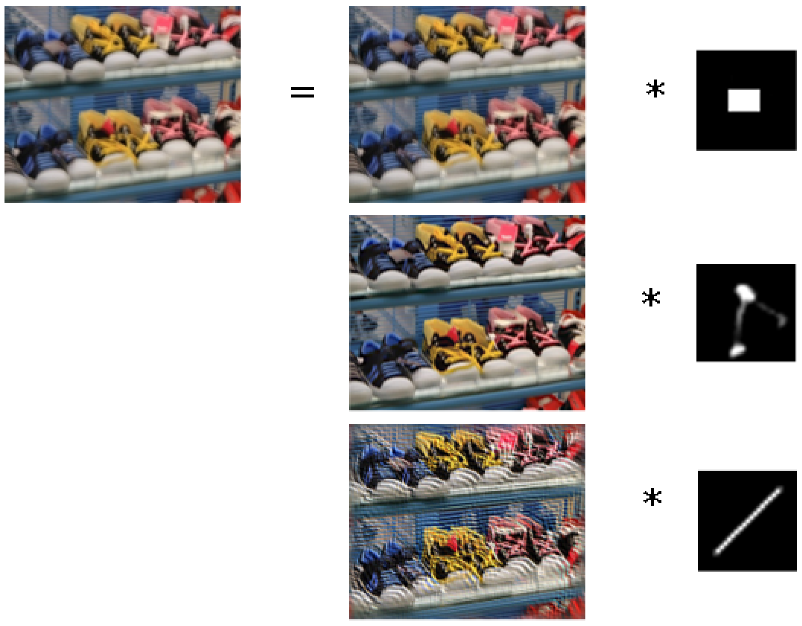


Figure 6. The deblurring process is an ill-posed problem since there are many possible solutions for the blurred image on the left [24].

Image deblurring has been studied extensively in the literature since a long time ago [25], and there are a huge variety of ideas that were proposed to tackle the problem. Since we propose a deep learning-based technique in this thesis, we refer to the non-deep learning-based techniques as "conventional" to distinguish them from the deep learning-based techniques. In this chapter, our goal is not to list the conventional techniques, but instead to introduce the essence of how the conventional methods approached the problem, and the bases that the devised algorithms and ideas are built on.

3.2. Uniform Deblurring

In uniform deblurring, the PSF function is assumed to be uniform for all pixels in the image. In this section, we discuss the commonly used approaches, which are Maximum A Posteriori and Variational Marginalization approaches.

3.2.1. Maximum A Posteriori

The most commonly used conventional model for image deblurring under uniform blur is the Maximum A Posteriori (MAP) which is based on Bayesian inference. The famous Bayes' rule is expressed as the following:

$$p(A|D) = \frac{p(D|A)p(A)}{p(D)} \tag{7}$$

where A, D are events, and $p(A|D)$ is the probability of event A occurring given that event D happened, and it is called the posterior probability. $p(D|A)$ is the probability of event D occurring given that A happened, and it is called the likelihood. $p(A)$ and $p(D)$ are the probabilities that event A and D happens respectively. $p(A)$ is known as the prior probability.

In our context and supposing blind deblurring, A represents the latent sharp image I and the blur kernel S, and D represents the blurry image B. $p(I)$ and $p(S)$ are the knowledge priors on the latent image and the blur kernel respectively. This can be expressed from (7) as follows:

$$p(I, S|B) = p(B|I, S)p(I)p(S)/p(B) \tag{8}$$

Since $p(B)$ is not changing and constant, it is more convenient to discard it and rewrite (8) as follows:

$$p(I, S|B) \propto p(B|I, S)p(I)p(S) \tag{9}$$

If S is known, i.e. non-blind deblurring, (9) becomes:

$$p(I|S, B) \propto p(B, S|I)p(I) \tag{10}$$

To explore how those terms can be further extended to find the latent sharp image and the blur kernel, we follow Shan et al. [8] work which proposed a blind image deblurring for uniform blur. The aim of this is to give you the sense of the complexity of the used optimization techniques which require heavy computation usually, and how the conventional methods addressed the image deblurring problem in general. If the image deblurring process is complicated for the uniform blur, it is intuitive that it would be much more complicated in case of the non-uniform blur. Interested readers are encouraged to check the full details in [8].

The likelihood $p(B|I, S)$ is based on the uniform convolutional blur in Equation (2), $I * S - B = n$. This term is usually called the data fitting term. However, this term is ill-posed and regularizers and/or constraints are needed. They proposed to model the likelihood as a set of independent and identically distributed (i.i.d) random variables for all pixels, where each of them follows a Gaussian distribution G. To consider the spatial randomness of the noise, the first and second order derivatives of the pixel noise are included in the likelihood. Let's refer to the set of used partial derivative operators as $Z = \{\partial_0, \partial_x, \partial_y, \partial_{xx}, \partial_{yy}, \partial_{xy}\}$, and ∂_* is an element in Z. The ∂_0 operator simply means no partial differentiation. The likelihood can be defined as follows:

$$\begin{aligned} p(B|I,S) &= \prod_{\partial_* \in Z} \prod_i G(\partial n_i | 0, \sigma_{\partial_*}) \\ &= \prod_{\partial_* \in Z} \prod_i G(\partial I_i | I_i^c, \sigma_{\partial_*}) \end{aligned} \tag{11}$$

where i is the pixel index, I_i is the pixel value in image I, I_i^c is the pixel value in the reconvolved image $I^c = I * S$, and σ_{∂_*} is the standard deviation (SD) when using the partial operator ∂_* .

The kernel prior $p(S)$ is based on the sparsity observation. This means that most of the blur kernel values tend to be zero, and only a small portion of the kernel values indicate how a point in the image moves. In addition, the values are assumed to be non-negative. This can be modelled by an exponential distribution as follows:

$$p(S) = \prod_j e^{\lambda S_j} \tag{12}$$

where λ is the rate parameter of the exponential distribution, and j indexes over elements in the blur kernel S. For the latent image prior $p(I)$, Shan et al. [8] introduced two components, local prior $pl(I)$ and global prior $pg(I)$ such that $p(I) = pl(I)pg(I)$. The global prior $pg(I)$ is based on the learnt gradients from natural images which follow a heavy-tailed distribution. To approximate the shape of the logarithmic gradient distribution, a piece-wise continuous function $\omega(\partial_* I)$ is designed. The $pg(I)$ component then can be defined as the following:

$$pg(I) \propto \prod_i e^{\omega(\partial_* I_i)} \tag{13}$$

The local prior $pl(I)$ is devised to avoid ringing artifacts which could be dark and light ripples that appear near strong edges after deconvolution. To handle this, the standard deviation of pixels' intensities are calculated inside a window of the same size as the blur kernel with the central pixel representing the local window. Each local window then is thresholded. If the value of the local window satisfies the threshold, the central pixel is included in a set Ω. The values of pixels in Ω are constrained such that their blurred gradients are close to those of the latent image gradients:

$$pl(I) = \prod_{i \in \Omega} G(\partial_x I_i - \partial_x Bi | 0, \sigma_1) G(\partial_y I_i - \partial_y Bi | 0, \sigma_1) \tag{14}$$

where σ_1 is the standard deviation. By substituting (11), (12), (13), and (14) into (9), and taking the negative log of (9), the maximization of the posterior becomes a minimization problem which is commonly called an energy function:

$$\begin{aligned} E(I,S) \propto & \left(\sum_{\partial_* \in Z} \lambda_1 \left\| \partial_* I * S - \partial_* B \right\|_2^2 \right) \\ & + \lambda_2 \left\| \omega(\partial_x I) + \omega(\partial_y I) \right\|_1 \\ & + \lambda_3 \left(\left\| \partial_x I - \partial_x B \right\|_2^2 \odot M + \left\| \partial_y I - \partial_y B \right\|_2^2 \odot M \right) + \left\| S \right\|_1 \end{aligned} \tag{15}$$

where λ_1, λ_2, and λ_3 are parameters that can be derived from the distributions. M is a 2D binary mask to select the pixels that exist in region ω, $\odot$ is an element-wise operator and $\|,\|_p$ denotes the p-norm operator.

This form of the energy function, with the data fitting term and the knowledge priors, is usually used in the conventional deblurring methods. A high abstract general form of the energy function can be formulated as the following:

$$E(I,S) = E_{fit}(I,S) + E_{reg}(I,S) \tag{16}$$

where E_{fit} is the data fitting term and E_{reg} is the regularization terms.

The function in (15) is very complex with a very huge number of unknowns, and requires sophisticated optimization methods. They solve the energy function iteratively by alternating optimization, i.e. fixing the blur kernel to estimate the latent image at first, then use the estimated latent image to estimate the blur kernel. In the beginning, the blur kernel can be initialized with a rough kernel estimate e.g. a straight line. Additionally, they leverage Fourier transformation to speed up the optimization process.

3.2.2. Variational Marginalization

Marginalization approach attempts to approximate the posterior distribution for both the kernel and the latent sharp image $p(I,S|B)$ by a simpler, factorized distribution using a variational method. The factorized form of this distribution means that it is straightforward to marginalize over the solution space of the sharp image in order to produce an estimate of the blur kernel S [3, 6]. While MAP-based approaches might converge to wrong solution, the marginalization-based approaches are more robust but slow [24].

Detailed analysis and evaluation of the MAP and marginalization problems in uniform motion deblurring is provided in [26]. In [27] , they introduce a comparative study of 12 uniform deblurring algorithms.

3.3. Non-Uniform Deblurring

The real world blur is often non-uniform. Camera rotations usually introduce spatially variant blur. The scene depth variation can also cause blur variations when the camera undergoes translations as discussed in Section 2.2.2. Methods dealing with non-uniform blur can be classified into two categories based on the used blur model: homography-based models and patch-based models. Since we are interested in exploiting the IMU measurements and the scene depth map to guide the deblurring process, we spotlight on a closely related work in the last two sections.

3.3.1. Homography-Based Models

The methods of this group model the non-uniform blur as an integration of the intermediate frames captured by the camera along its motion trajectory. A global descriptor can be provided by a set of camera poses during the capturing process. Apart from the different ways in which the methods tackled the non-uniform image deblurring, the underlying idea is quite similar to the homography-based modelling discussed in Section 2.2.4.

For instance, Whyte et al. [3] proposed a homography-based model for non-uniform blur. Supposing mainly small camera rotations and translations, they showed that the effect of the rotations is significantly larger than translations. Thus, they considered that most of the blur is attributed to 3D camera rotations around its optical center (i.e. in/out-of-plane rotations). To adapt uniform deblurring methods for non-uniform deblurring, they substituted their model into two blind uniform deblurring methods [6, 9], and another method that uses noisy/blurry image pairs [22]. Gupta et al. [28] modelled the non-uniform blur in a similar approach by introducing a motion density function, which finds weights for the whole possible camera poses. However. They only considered in-plane rotations and translations (i.e. 3 degrees of freedom motion). Tai et al. [4] modified a popular conventional non-blind uniform deblurring algorithm Richardson-Lucy [25, 29] to handle non-uniform deblurring by incorporating a projective motion path blur model (i.e. a homography-based model). One problem when adapting a pure homography-based model in the conventional methods is the difficulty of incorporating fast Fourier transform to speed up the optimization process.

3.3.2. Patch-Based Models

The non-uniform blur is tackled by partitioning the image into patches. Every batch is assumed to have an invariant blur kernel, as illustrated in Figure 7. This can be mathematically described by slightly modifying (4):

$$B = \sum_{r} S^{(r)} * (w^{r} \odot I) + n \tag{17}$$

where $S^{(r)}$ is the blur kernel at image region r, and $w^{(r)}$ is an image of weights with the same dimension as the latent image I, such that the pixels in region r can be expressed as $(w^r \odot I)$ [16].

In [13], they build a tree structure for the image regions based on a depth map and apply a coarse-to-fine kernel estimation from the top level regions to the leaf level regions.

In contrast to the homography-based models, the patch-based models do not provide a global descriptor for the blur. However, a patch-based model can take advantage of using Fourier transforms to speed up the optimization process. Hirsch et al. [30] and Hu et al. [16] use a hybrid model where a homography-based model is used for the blur kernels. This can be achieved by modelling the $S^{(r)}$ as a linear combination of kernel bases H_θ, such that $S^{(r)} = \sum_\theta \mu_\theta H_\theta$ where H_θ is the homography at camera pose θ and defined in Equation (6). The variables μ are the coefficients of the kernel bases, which are determined by the camera rotational and translational movement [16]. Furthermore, the coefficient μ_θ can be considered as the weight equivalent to the portion of the time camera spends at pose θ during the exposure time.

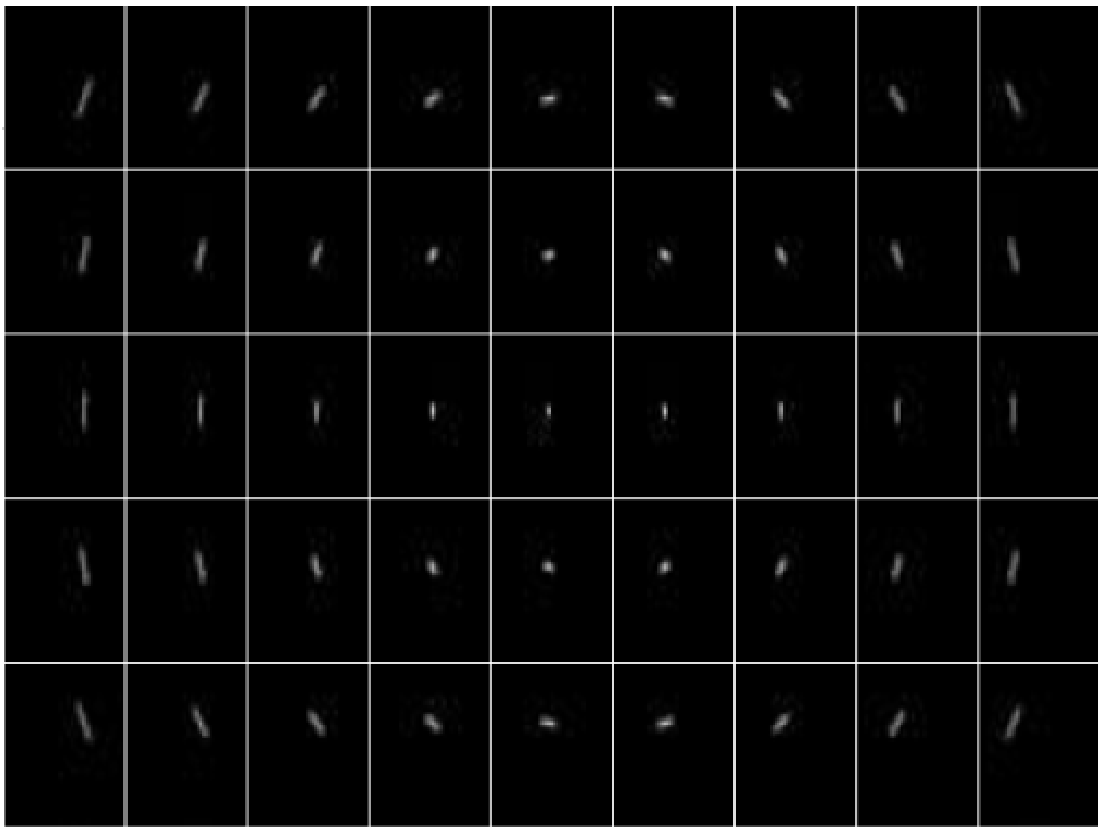


Figure 7. An image, with spatially-variant blur, is partitioned to small patches, such that the blur in every patch is almost uniform.

3.3.3. IMU-Aided Methods

In this section, we talk about the methods that leveraged the IMU measurements to estimate the blur kernel to be used in image deblurring process. Nowadays, most of the modern mobile devices and handheld cameras are equipped with IMU unit. The IMU unit usually contains a gyroscope and an accelerometer. A gyroscope measures the angular velocity at a given point around an axis. It can be used to measure the rotation around an axis. An accelerometer measures the acceleration (i.e. rate of change in velocity) at a given point along an axis. It can be used to measure translation from an initial position.

Joshi et al. [15] attached special hardware equipped with IMU to a consumer camera. They used the accelerometer and the gyroscope to estimate 6 degrees of freedom rotations and translations during the exposure time. Those 6D poses then are substituted into a homography-based model to estimate the blur kernels. To recover a sharp image using the estimated blur kernel, they applied their blur model into a MAP framework in (9) with a hyper-Laplacian image prior, and an energy function was formulated by taking the negative log of the posterior. The IMU measurements are prone to high error due to the integration of noisy sensor data which is known as drift. The integration of noisy data can cause temporally increasing deviation from the real motion. The noise from the accelerometer is worse since the noisy data is integrated twice to get the position. To alleviate this problem, they introduced a drift compensation technique inside the MAP framework. They showed the effectiveness of adapting the image prior and the drift compensation instead of using the IMU measurement directly.

Hu et al. [16] analyzed the practical issues of using smartphone IMUs for kernel estimation. The IMUs used in smartphones are of less quality in order to limit power consumption. They discussed the following issues:

- Time delay: there is a problem of synchronization between sensors since they have different warm-up times.
- Rotation center: in [15], the rotational center is assumed to be in the optical center, which is not always the case. Furthermore, the rotation center might differ during a single capture depending on the exposure time length.
- Noisy sensor data: this is the same as the drift issue mentioned in [15]. The noise in a smartphone IMU is expected to be worse.
- Gravity effect: the contribution of the gravity to the estimated translations is unknown since it is difficult to estimate the initial orientation and velocity of the camera.

In addition to these issues, the scene depth variation can affect the validity of the estimated blur kernel when it depends only on the IMU measurements. To address this, they exploited the phase-based auto-focus built into most smartphones to provide sparse depth information. They proposed an image-based online calibration to address the time delay and rotational center issue. The light streaks in the scene are exploited whenever available. If there are no light streaks detected in the scene, they use a power spectrum analysis technique. They adopted a region-based blur model and applied a homography-based model for the kernel estimation inside each region. An energy function is formulated which is similar in spirit to (16). To minimize the energy function, they applied an alternating optimization technique. At the beginning, the kernel is initialised using the reading from the sensors, then a coarse-to-fine manner is followed for kernel refinement. Finally, they noticed that their proposed method has a limitation with the shallow scene depth, e.g. less than 0.5 meter, and the camera undergoes a significant translation. In this situation, the blur kernel estimated initially from the IMU readings can significantly differ from the true blur kernels.

3.3.4. Depth-Aided Methods

As discussed in Section 2.2.2, the scene depth variation can cause non-uniform blur when the camera undergoes translational movement. Thus, leveraging the information of the scene depth can be critical for estimating the blur kernel. In this section, we mention two methods that use the scene depth information to guide the image deblurring process.

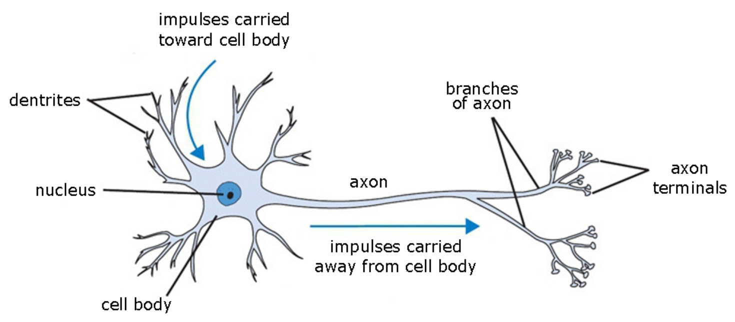
Xu et al. [13] adopted pure patch-based model. Instead of simple partitioning of the image into small regions, a tree-structure is built for the image regions based on a depth map. The first step in their algorithm is to find a disparity map by stereo matching for a blurred stereo image pair. After constructing the region-tree, the blur kernels are estimated from high level nodes and propagated to lower level nodes for blur kernel refinement. To estimate the blur kernel and the latent image, they apply alternating optimization of an energy function similar to (16). In an iterative manner, the estimated latent image is then used to refine the disparity map and rebuild the region tree, and the previous steps are repeated.

Pan et al. [14] proposed to jointly estimate the camera motion and remove the non-uniform blur from a single image using a depth map. They considered 6 degrees of freedom camera motion and supposed small camera motions. A homography-based model is adopted for the blur kernel, which contains the scene depth as an argument similar to (4). They formulate an energy function with a data fitting term and carefully designed regularization terms, which is similar to (16). To minimize the energy function, they apply alternating optimization. The latent image is fixed to estimate the motion variables, and then the estimated motion variable is used to estimate the latent image. The alternation continues iteratively till convergence.

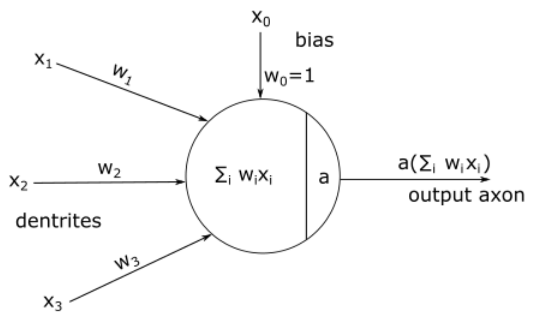
4. DEEP LEARNING-BASED IMAGE DEBLURRING

4.1. Deep Learning and Neural Networks

Deep Learning is a subset of machine learning algorithms. As it is well known, machine learning aims to generalize to unseen data by learning from data samples called training dataset. Recently, Deep learning, especially DNN models, lead to a breakthrough in a wide domain of challenging image processing and computer vision tasks. In this chapter, we give a brief introduction about deep neural networks, especially convolutional neural networks and its U-net architecture. We assume that the reader has some knowledge of machine learning and deep learning. However, some basic concepts and definitions are introduced as long as it is convenient.


impulses carried toward cell body
dentrites
branches of axon
axon
axon terminals
nucleus
impulses carried away from cell body
cell body

(a) A biological neuron [31].



(b) An artificial neuron. x_1, x_2, x_3 are the input features, x_0 is the bias, and w_i are the weights.

Figure 8. The biological neuron and the artificial neuron.

The neural network is one of the approaches used in machine learning. The elementary computational unit in the neural network is the neuron or the node. In general, the neural network is inspired from the biological neural system. Figure 8 shows an artificial neuron and a real biological neuron. The inputs to the neuron are the input features or the outputs of other neurons multiplied by weights, and a bias. The neuron sums these inputs together and fires a signal based on an activation function. The activation function is a non-linear function that models whether the neuron is activated based on the inputs. This simple neuron is also known as Perceptron. A multi-layer perceptron (MLP), or simply an artificial neural network consists of an input layer (data inputs or features), an output layer, and a number of hidden layers. The word *artificial* is sometimes used with neural networks to distinguish them from the biological neurons. However, neural networks usually refer to artificial neural networks. A typical neural network with one output is depicted in Figure 9 . A loss function (or a cost function) is designed to encourage the predicted output of the network to follow the ground truth values (*i.e.* the distribution of the training data). To avoid model overfitting on the training data and help the model generalize well to unseen data, several regularization techniques are applied such as weight decay, dropout rate, batch normalization, noise addition, dataset augmentation, and parameter sharing [32]. Some of those techniques are not used directly for regularization, but they introduce a regularization effect.

To find the parameters (*i.e.* the weights and the biases) of the network, an optimization technique for the cost function is used to update the parameters with a learning rate. The backpropagation algorithm [33] is used to compute the gradients of the cost function and propagate the computed gradients backward in the network by recursive application of the chain rule of calculus to calculate the gradients in each layer. The calculated gradients in each layer are then utilized by the optimizer to update the parameters. The training process can be considered as iterative output (forward pass) and feedback (backward pass) process. An example of the loss function is the mean squared error (MSE), also known as L2-norm, between the reference output and the predicted output of the network. One popular optimizer used frequently for training DNN is ADAM optimizer [34].

An essential difference that distinguish the deep learning techniques from other machine learning techniques is that the input features are not hand crafted, but they are designed automatically by the model. This means that the model will learn those features. This is known as representation learning [32]. Furthermore, the deep learning model can learn more abstract representation from the simpler representation. For instance, if some shallow layer learn some primitive shapes, the deeper layer can learn more complex shapes from those primitive shapes. Figure 10 depicts this difference. Theoretically, the universal approximation theorem [32] states that a feed forward network can represent any function.

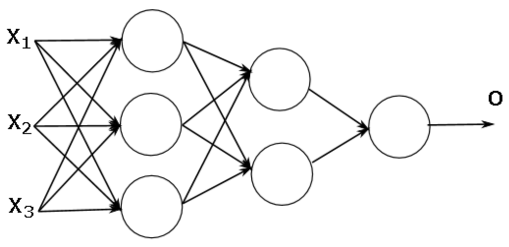


Figure 9. A typical neural network example of three inputs in the input layer, two hidden layers, and one output in the output layer.

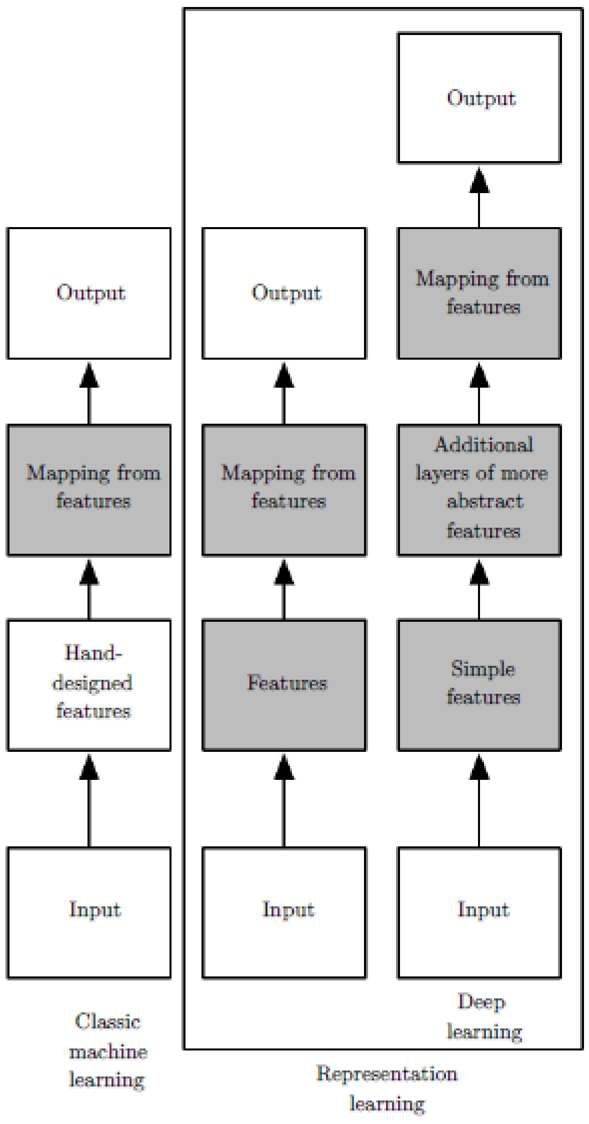


Figure 10. A flow chart depicts how deep learning techniques differ from other machine learning techniques [32].

4.2. Convolutional Neural Networks

A convolutional neural network (CNN) [35, 36] is a variation of deep neural networks. While CNNs have already existed for a long time, their success was limited due to the size of the available training datasets and the size of the considered networks [37]. Recently, the usage of the CNNs led to tremendous advancements in a broad domain of computer vision and image processing tasks. A popular architecture of CNN consists of three basic components: a convolution layer, pooling layer and fully connected layer. This architecture is suitable for image classification tasks. The fully connected layer is added after multiple downsampling of the input. A fully connected layer means that every node in the layer is connected to every node in the previous layer. In classification tasks, nonlinear activation function (*e.g.* softmax) is used at the final layer to compute the class scores, which is different from the nonlinear activation function used after each convolution layer (*e.g.* rectified linear unit or ReLU [38]). A more abstract architecture of CNN is illustrated in Figure 11, which depicts the CNN as cascading of blocks that comprise a convolution layer, nonlinear activation function layer, and pooling layer.

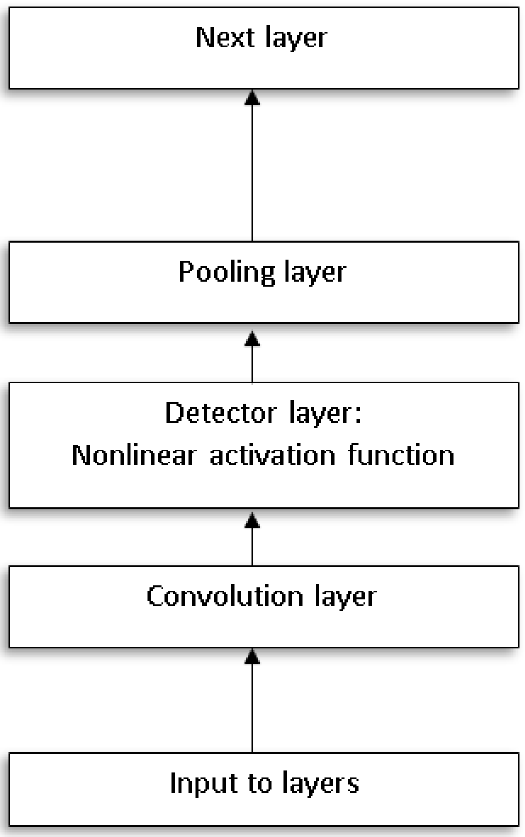


Figure 11. An abstract architecture of a typical CNN as cascading blocks of convolution, nonlinear activation, and pooling layers [32].

4.2.1. Convolution Layer

The convolution is a linear operation which is similar to the conventional convolution used in image processing for image filtering (discrete convolution). The input is convolved by a so-called filter or a kernel. The output of the convolution is usually called *the feature map*. Since we are dealing with images, the most common convolution operation is the 2D convolution. Figure 12 illustrates a 2D convolution example. In 2D convolution, the filter has the same depth (number of channels) as the input. The number of output channels is determined based on the number of used filters. The output size (*i.e.* width and height) is determined by the kernel size, strides, and padding. The stride is the distance between two consecutive application of the filter along one direction. Padding is usually applied by inserting additional zeros into the input. For instance, one can add the necessary padding values to keep the output size the same as the input. The output size can be determined using the following equation:

$$od = \frac{iz - ks + 2 * pd}{st} + 1 \tag{18}$$

where od, iz, ks, pd, and st are the output size, input size, kernel size, and padding, respectively.

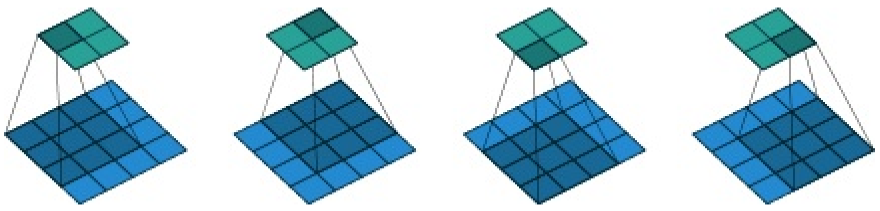


Figure 12. An example of 2D convolution with 2*2 kernel over 4*4 input with unit strides, and no padding [39].

Convolutional neural networks exploit three interesting ideas *sparse interaction*, *parameter sharing*, and *equivariant representation* [32]. For sparse interaction (also known as local connectivity or sparse weights), each entry in the 3D volume output (the feature map) can be interpreted as an output of a neuron that looks into a local region of the input as illustrated in Figure 13. This local region is referred to as the receptive field, which is the same as the kernel size in this case, *i.e.* its height and width. The receptive field of the nodes in a deeper layer in the CNN is larger since they would look indirectly into a spatially larger region of that input. Figure 14 depicts the receptive field of a shallow layer and a deeper layer. Instead of the nodes in a layer connecting to every node in the previous layer similarly to fully connected nodes in the neural network in Figure 9, the nodes connect only to a small number of nodes in the input. For instance, if we need to detect small and meaningful features in the image, *e.g.* edges, then those features are placed in a small patch of pixels inside the whole image. Moreover, the neurons of one channel in the output can share the same set of parameters. This means that one channel of the output in Figure 13 is a result of applying the same kernel. This is applicable because if we have some parameters that detect a specific type of feature in the image, there is no need to learn a new set of parameters to detect the same feature in other regions of the image. Consequently, the convolution becomes equivariant to translations (i.e translationally-invariant). This means that if the input is translated, the result of the convolution would be translated as well. However, the convolution is not naturally equivariant to some other transformations, such as changes in the scale or rotation of an image [32].

The parameter sharing scheme introduced by the convolution layer decreases significantly the number of model parameters, which in turn decreases the needed memory to save the model parameters. For example, in Figure 13, let us assume that we have an input with a size $32 * 32 * 3$ features (*i.e.* $iz = 32$), and 5 filters of size $4 * 4$ (*i.e.* $ks = 4$) with a stride $st = 2$ and no padding. By applying the equation (18), the output volume size would be $15 * 15 * 5$. If we considered the fully connected neurons scheme, the parameters (weights and biases) for this layer only shall be $(32*32*3)*(15*15*5) = 3,456,000$ weights, in addition to $15*15*5$ biases which would result in $3,457,125$ parameters in total. Considering the local connectivity of

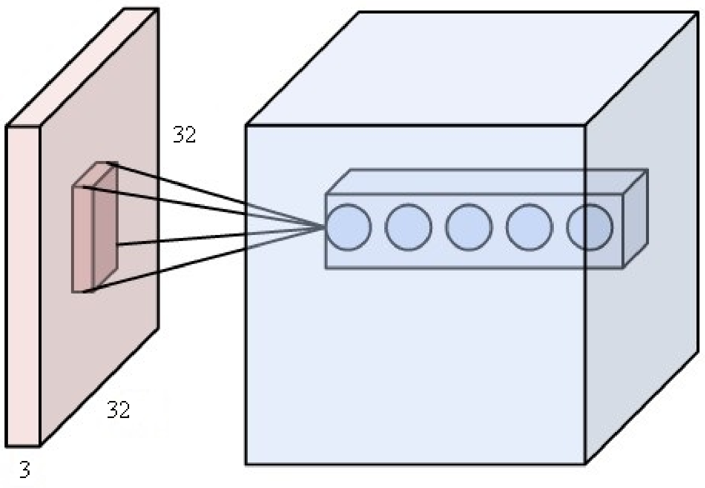


Figure 13. The convolution output (on the right) can be interpreted as a 3D volume of neurons, such that each neuron looks into a local region of the input (on the left) [31].

the neurons (*i.e.* the sparse interaction property), each neuron then looks into a local region with size $4 * 4 * 3$, so each neuron has $4 * 4 * 3 = 48$ weights plus one bias (*i.e.* 49 parameters in total). This would result in $15 * 15 * 5 * 49 = 55,125$ parameters. Finally, considering the parameter sharing scheme in addition to local connectivity, each neuron in a channel in the output volume shares the same parameters (*i.e.* uses the same filter). Thus, we shall have $(4 * 4 * 3 + 1) * 8 = 392$ parameters only. There are some applications where assuming parameter sharing scheme would not be valid. Consequently, the parameter sharing scheme is relaxed to local connectivity. The weights and biases in the convolution layer are what the CNN learns during the network training phase.

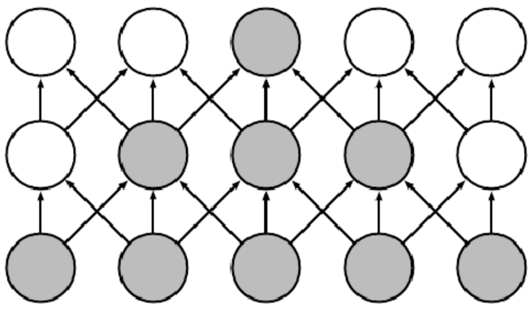


Figure 14. The deeper layers have larger receptive field than the shallower layers. For instance, the shaded unit in the top row looks indirectly into all units in the bottom row, while the shaded units in the middle row look into three of them each [32].

4.2.2. Dilation

Dilation [40], also known as atrous (a french word means *with holes*), refers to that the filter can have dilation or spacing between its parameters along one dimension. This can be explained by Figure 15. Therefore, we can modify equation (18) to properly calculate the output size considering the kernel dilations as the following: $od = (iz - dl * (ks - 1) + 2 * pd - 1)/st + 1$, where dl is the dilation rate. Dilations can increase the spatial extent of what filters can process without introducing additional parameters.

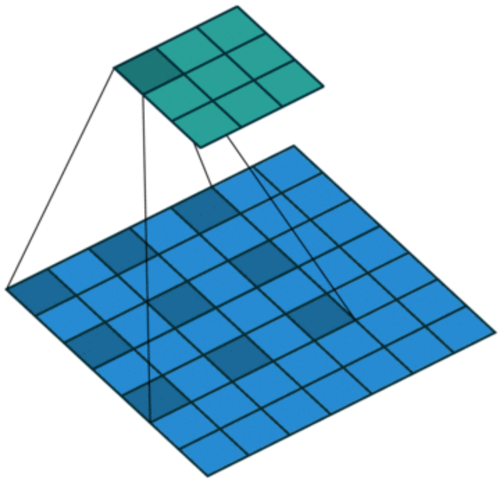


Figure 15. A dilated convolution example where a 3*3 kernel is convolving over a 7*7 input with a dilation rate of 2 [39].

4.2.3. 1×1 *Convolution*

This type of convolution layer was first investigated by Network in Network [41]. The usage of 1*1 convolution would make no sense at first since it can be considered simply as a scaling operation. However, when taking into account that we deal with 3D volumes, such that the input is processed along its depth (*i.e.* the channels) and the depth of the filter extends to the full depth of the input, the 1*1 convolution layer then introduces a means of fusing features along multiple channels.

4.2.4. Deformable Convolution

In the deformable convolution [42], the kernels no longer have a regular rectangular shape, but they can be of any arbitrary shape. The offsets of the kernel parameters from its rectangular shape are learned during the network training. The usage of the deformable convolution extends the spatial capability of the convolution layer to get information that is not organized in a regular shape within the input.

4.2.5. Transpose Convolution

Transpose convolution, also known as deconvolution or fractional stride convolution, can be considered as the inverse operation of convolution as depicted in Figure 16. For instance, one can use transpose convolution to go from something that has the shape of the output of convolution to something that has the shape of its input. That is the case when using the deconvolution in U-nets discussed in Section 4.2.8.

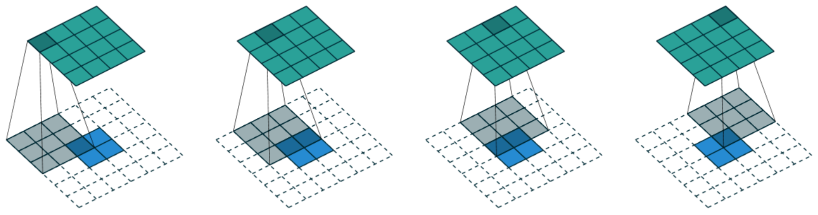


Figure 16. A transpose convolution example. It is equivalent to convolving a 3*3 kernel over a 2*2 input with a unit strides and 2*2 zero padding [39].

4.2.6. Pooling Layer

A pooling layer is added to downsample the input. It is usually added after performing the convolution and the activation. From its name, it can be considered that the pooling layer summarizes and pools the input for some existing feature disregarding its exact position. In other words, it can be shown that pooling is invariant to small translations in the input. This means that if we translate the input by a small amount, most of the pooling output values do not change [32]. One popular method is max pooling which is usually performed spatially on a 2*2 region (or generally a small region) and outputs the maximum value in that region. There are other popular variations of the pooling such as average pooling, L2 norm pooling, and weighted average pooling based on the distance to the centre pixel. The pooling layer performs a fixed operation, so there are no parameters to be learned here.

4.2.7. Residual Block

The residual block was first introduced in [43]. The usage of residual blocks enabled building deeper networks for better higher-level representations learning. The ability to add more layers to make the network deeper was hindered by the problem of vanishing gradients. It was observed that the gradients vanish (*i.e.* fall to zero) when adding more layers to the network. Besides, adding more layers makes the training error higher. They addressed this problem by proposing the residual block depicted in Figure 17. The residual block contains a shortcut connection (also known as a skip connection) which skips one or more layers to add the identity mapping of the input to the output result of those stacked layers. In some contexts, the skip connection is sometimes used to indicate a concatenation operation, while residual connection indicates to an addition operation.

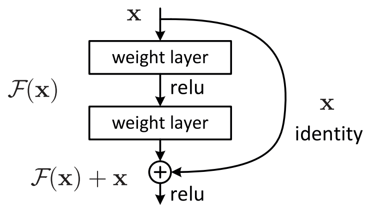

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 17. A residual block [43].

4.2.8. U-Net Architecture

The U-Net was first coined by Ronneberger et al. [37] for biomedical image segmentation. The U-Net architecture in Figure 18 consists of contracting path and expanding path. The inputs are downsampled by convolution and pooling till reaching a bottleneck after which the inputs are upsampled using deconvolutions. The U-Net architecture introduces a per-pixel output which is useful in many image translation tasks [44] and image restoration. While the architecture of the U-Net is basically similar to the pure encoder-decoder network architecture, the main difference is the usage of the skip connections between the encoder layers and decoder layers. For instance, in Figure 18, the outputs in the encoder part are concatenated with the outputs from the decoder parts. In the pure encoder-decoder, the inputs are encoded into some representation or codes, which are then used by the decoder to reconstruct the output without any other information from earlier layers before the bottleneck. Hence, the training and the usage of encoder and decoder part can be performed separately, while this is not the case in U-Net where there are connections between the encoder and the decoder.

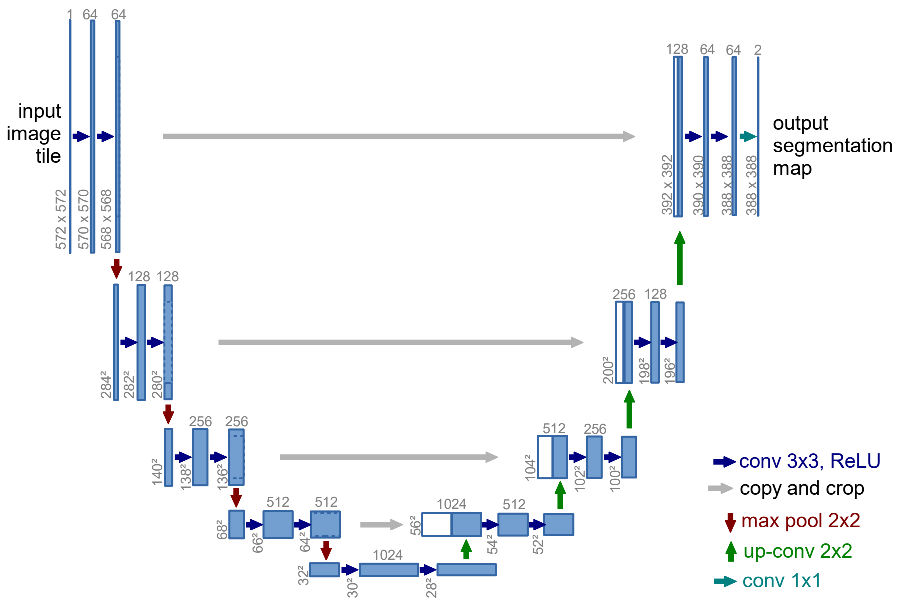


Figure 18. The U-Net architecture example from [37]. Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The width and height are provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

4.3. Deblurring Techniques

With the recent success of DNNs in many tasks, it was attempting to leverage them in low-level computer vision tasks such as image deblurring. In Chapter 3, we discussed how the conventional methods approached the image deblurring problem through designing of complicated energy functions. These functions are then solved using sophisticated optimization techniques that usually require heavy computations and take a long time. For instance, Whyte et al. [3] report that their adopted non-uniform deblurring algorithm can take several hours to deblur an image of several hundred pixels across on a modern workstation. Moreover, they assume only small 3D camera rotations around its optical centre which are supposed to be the typical handheld camera shakes. In addition to that, conventional methods are still suffering when handling severe spatially variant blurred images arisen from different and various camera motions, in addition to scene depth variation as well. On the other hand, and with the advances in the deep learning techniques and DNN architectures, the DNN showed interesting and promising results in image deblurring in terms of speed and output quality [45, 46, 47, 48, 49, 50]. However, there are challenges to be addressed, namely, the dataset availability and the hyperparameters of the learning model.

Dataset Availability: Simultaneously acquiring real blurred and sharp image pairs is extremely challenging. However, we show a promising setting of acquiring such data in Chapter 5. Therefore, in most of the cases, the training dataset is synthesized [50, 49]. In this context, we also show a novel way of synthesizing depth-dependent blurred images in Chapter 5. Moreover, the success of deep learning based approaches depends highly on the statistical consistency between the training dataset and the real world blur cases. Hence, the training data should provide many variations of scenes and camera motions to enable the learning model to generalize well to the unseen real world blur cases.

Model Hyperparameters: Any parameter that is not learned during the training phase of the DNN model is called a hyperparameter. There are many model hyperparameters such as strides, dilations, padding, number of channels and kernel size with convolution layers, number of layers, learning rate, regularization parameters, dropout rate, batch normalization existence, weight decay parameters, etc. There are many rules of thumb for finetuning hyperparameter. Furthermore, there are optimization methods for finding suitable hyperparameters [32]. The model architecture itself can be considered as a hyperparameter. The network architecture and components required for addressing the image deblurring are still subject for broad improvements.

In this section, we introduce learning-based approaches for image deblurring and then talk about closely related work, namely, DeepGyro [50] and DAVANET [49].

By considering the image deblurring process as two joint steps: the kernel estimation step and the the deconvolution step, we can learn any of them separately or directly output the latent sharp image from the blurred input. Sun et al. [51] adopted a classification CNN to predict sparse blur kernels for a local patch. A dense motion field is obtained via Markov Random Fields (MRF) from the sparse blur kernel. The final latent sharp image is then recovered by a conventional non-blind deblurring method. Gong et al. [52] utilized a fully CNN to estimate the dense motion flow from the blurred image and used the same conventional method to recover the latent sharp image. Xu et al. [53] employed a deep CNN for sharp edge predictions, which is then used to estimate the kernel and recover the sharp image using a conventional deblurring method. In [54], they used a neural network for predicting Fourier coefficients for a deconvolution filter. In [55], they partitioned the deblurring process to three modules: feature extraction module, kernel estimation module and image estimation module. Through the modules, they utilize deep neural networks and add other layers and components that are inspired from the conventional techniques. They iterate over the three stacked modules to recover the latent sharp image. They also mentioned other previous usages of neural networks in the literature. Others [56, 57, 58] trained a DNN for non-blind deconvolution to avoid the artefacts that are usually created by the traditional non-blind deconvolution methods.

Instead of learning the kernel estimation and non-blind deconvolution separately or replacing some steps in the conventional techniques with learning-based methods, many single image deblurring methods [47, 59, 46, 48, 45, 50] were proposed to output the latent image directly from the blurred one using a CNN-based solution. To increase the receptive field and handle different blur kernel sizes, [47, 46, 60, 59] proposed multi-scale system in a coarse-to-fine manner. This means that they gradually restore the latent sharp image on different image resolutions similar to a pyramid scheme [46]. In [47], they proposed a multi-scale fully CNN architecture for image deblurring such that the deblurred low-resolution image is given to the next finer scale after upsampling by upconvolution layer to match the next scale input size. Their model includes the residual block as a building block. In a similar way, [46] designed a multi-scale recurrent neural network (RNN). Each scale is an encoder-decoder CNN with skip connections between the corresponding layers in the encoder and decoder, and residual building blocks. Different from [47], they share the parameters among the three scales and use a Long-Short Term Memory (LSTM) unit to share information across scales. In addition, they use bilinear interpolation for upsampling (i.e. no parameters are learned). Instead of no-sharing or sharing of parameters across the scales, [60] proposed parameter selective sharing across the scales. They also introduce the usage of nested skip connections inside each encoder-decoder CNN scale instead of simple skip connections. In [59], they followed the coarse-to-fine fashion but with three different fully convolutional subnetworks. Together, those subnetworks form an architecture similar in spirit to an encode-decoder. Each subnetwork includes a residual connection with the output. To handle spatially variant blur in dynamic scenes, [48] proposed a spatial variant RNN. A decoder-encoder-style fully CNN with skip connections is used to learn spatially variant weights for a four RNNs. Two other fully CNNs, with skip connections between them, are used for feature extraction for the RNNs, and image construction from the output of the RNNs. Bilinear interpolation was used for upsampling operations to avoid artefacts created by deconvolution layers. In [45], they proposed a conditional generative adversarial network (GAN) for blind motion deblurring. GANs [61] include two networks playing against each other: a generator and a discriminator. In our context, the generator tries to output a latent sharp image, while the discriminator tries to decide whether this output is sharp. In other words, the generator tries to fool the discriminator. With this concept, they train a fully CNN and use residual building blocks that include an instance normalization layer [62]. The discriminator architecture is identical to PatchGAN [44]. Similarly,

[47] adopt adversarial loss by training a classification CNN as a discriminator. The adversarial loss is used to obtain realistic sharp results.

In addition to the single image deblurring, many multi-image deblurring methods were proposed. In principle, they take advantage of the different information that can be aggregated from multiple images. Mustanieni et al. [63] utilized U-Net-style fully CNN called LSD$_2$ for jointly denoising and deblurring a pair of long-exposure blurry and short-exposure noisy images. Furthermore, they introduced a novel technique to generate synthetic data for network training. Aittala et al. [64] proposed a fully CNN for recovering a latent sharp image from a burst of differently degraded images. To give equal consideration to all frames in the burst, they reflect the permutation invariance of the frames into the U-Net architecture by sharing pooling layers between the identical networks.

Moreover, video deblurring methods [65, 66, 67, 68, 69] can exploit additional spatial and temporal information that exists across neighbouring frames. However, in most cases, they should seek how to handle the alignment of neighbouring frames together. Some of the above methods were proposed originally for dynamic scene deblurring [47, 59, 46, 48, 60, 67, 68, 69], and some were not [50, 64, 63, 65].

4.3.1. IMU-Aided Deblurring

Mustaniemi et al. [50] proposed a gyroscope-aided image deblurring network called DeepGyro. It is assumed that the images suffer from spatially-variant blur resulted from small camera rotations around its optical centre which are supposed to be the typical handheld camera shakes. With this assumption, the effect of translation can be negligible, as shown in [3]. Furthermore, the blur kernels are supposed to be linear under relatively small exposure time assumption. The gyroscope measurements are leveraged to estimate the camera rotations. The blur kernels are then estimated using the planar homography-based model discussed in Section 3.3.1. They also considered the effect of the rolling shutter while computing the blur kernels. Modern mobiles are usually equipped with a rolling shutter. This means that the image rows are not read at the same time. Instead, each row is read at a slightly different time. If the camera is moving as well, this can cause edge tilting artefacts in the image. To make the network robust and able to generalize well to unseen blurry images, they add a slight noise into the exact estimated blur kernels. The noisy gyro-based estimated blur kernels and the blurry image are stacked together and fed to a U-net-style fully convolutional network.

To generate a training dataset, the homography model was exploited to calculate planar homographies of the sharp image using the sampled camera poses (*i.e.* rotations) during the exposure time. The calculated homographies are then averaged to synthesize the blurred image.

4.3.2. Depth-Aided Deblurring

Zhou et al. [49] proposed a unified framework for stereo image deblurring using the scene depth map and the aggregated information from the two views. The depth map is estimated from the stereo images using a disparity network which predicts the bidirectional disparity maps (*i.e.* left and right disparities). Each view is provided as an input to a U-Net-style fully CNN called DeblurNet. A fusion network is used to fuse the disparity information, the encoder feature maps and the two views. The fusion network consists of two small subnetworks called DepthAwareNet and GateNet. The disparity output is further fused with the second last layer features from the disparity network to produce fused depth features using DepthAwareNet. They benefit from the different view information existing in a pair of stereo images by fusing them together. To achieve this goal, the right view and its corresponding features generated from the encoder of DeblurNet are warped to the left view using the left disparity map to align them together. The two aligned views are fused together using GateNet. The output of the GateNet serves as a map for accepting the good features and rejecting incorrect ones. The two aligned features are fused together using the map produced by GateNet. Finally, the fused view features, the fused depth features and the original view features are concatenated together and fed to the decoder of DeblurNet. Those steps are then repeated for the right view.

To increase the receptive field and the spatial information for the deeper layers without increasing the model size, they concatenated four differently dilated convolution layer outputs together. They named this block as the context module. They showed the effectiveness of using the depth map and the context module on the image deblurring results.

For the DeblurNet loss, they utilized the perceptual loss [70], in addition to the MSE loss. The perceptual loss is defined as the L2-norm between the VGG-19 [71] features of the predicted image and the reference sharp image.

To generate a synthetic dataset, they simulate a long exposure by averaging neighbouring frames. They increased the frame rate by a frame interpolation method to avoid ghosting artefacts existed in a previous dataset called GoPro [47].

5. IMPLEMENTATION

In the previous chapters, we discussed how the conventional techniques and the deep learning-based techniques approached the problem of image deblurring. In this thesis, we propose a deep learning-based approach for image deblurring. The image deblurring process is guided using the depth map and the IMU measurements. Recently, mobile phones are equipped with multiple cameras for many purposes. One of those purposes is estimating the depth map using a stereo camera or multiple cameras. Although the range of the depth that can be captured by the cameras is limited by the very small distance (*i.e.* the baseline) between the cameras, the effect of the depth variations on the blur kernels decreases as the depth increases, and it requires an aggressive translation in the camera motion to appear, which is far from the typical camera shakes. In other words, the depth of the far objects has not to be very accurate for the image deblurring purpose. Moreover, some phones are equipped with a depth sensor. However, the quality of these sensors is limited due to power consumption considerations and the issues related to the technology behind these sensors. The modern phones and handheld cameras typically contain an IMU as well. The potential issues with these embedded IMUs were discussed in Section 3.3.3.

The main aspects of a deep learning-based approach are the training dataset and the model architecture. In this chapter, we introduce the details of a novel approach of synthesizing depth-dependent blurred dataset. The generated synthetic blur is spatially-variant due to the arbitrary rotations and translations of the camera and the scene depth variations. After that, we propose a novel and promising setup for acquiring real blurry and sharp pairs. While this is extremely challenging, we have passed a long track of development in that direction. Finally, we introduce our deep learning model architecture.

5.1. Synthetic Dataset Generation

One challenge of adopting a deep learning model is that it requires very huge dataset for training. Acquiring such enormous dataset in the real world is very exhausting. In this section, the novel technique provides an approach of efficiently synthesizing a huge number of synthetic blurred images using sampled camera poses and an RGB-D dataset.

5.1.1. Blur Generation Steps

As discussed in Section 2.2.2, the blur resulted from arbitrary camera motions (*i.e.* rotations and translations) is usually spatially variant (*i.e.* non-uniform). Moreover, the cases, where significant scene depth variations and camera translations exist, can produce depth-dependant non-uniform blur.

To generate a synthetically blurred image, we use sample camera motions (acquired using real camera motion) that simulate the arbitrary camera shakes. The camera motion is recorded using the IMU. The IMU's gyroscope and accelerometer measurements are leveraged to retrieve the camera rotations and translations using the methods shown in [50, 16]. In addition to the camera motions, the algorithm requires a sharp image and its corresponding per-pixel depth map, typically an RGB-D image.

The algorithm is based mainly on the blur homography-based model discussed in Section 3.3.1. Equation (4) reveals the needed arguments for calculating the homography, which are the rotation, translation, and the depth map. One more parameter is the camera intrinsic matrix which is set based on the sharp image specification. If no information is provided for the camera matrix, one can then choose the camera matrix parameters in (1) based on the image width and height in pixels. For instance, the principal point can be chosen to be the image centre, while the focal length can be close to or larger than the image dimension.

We have a number of randomly sampled camera poses during the exposure time τ. This number is determined by the sampling interval sf. For each camera pose, we calculate the planar homography. The homographies are calculated per pixel since we have different homography matrix for each pixel depending on its depth value. The new position of each pixel is then determined using Equation (5) based on the calculated homography for this pixel. The sharp image is then warped to that pose using the calculated homography for each pixel. Finally, we calculate a weighted average for all warped images during the exposure time using Equation (4). If we give equal weight to every pose, this becomes simply a summation of the warped images and then dividing by their number (which is also the number of sampled camera poses during this exposure time). The steps of the method are shown in Algorithm 1.

Algorithm 1. Synthetic Blurred Image Generation

```
   Input  : Sharp image $I$ , depth map $D$, camera poses $\theta$, sampling interval $sf$,
            and exposure time $\tau$
   Output: blurry image B
1  Sample a number of camera poses $\theta'$ using exposure time $\tau$ and sampling
     interval $sf$
2  Initialize a variable: $average = 0$
3  for each camera pose $\theta'_j$ in $\theta'$ do
4      Compute a homography for each pixel point $x$ in $I$ using Equation (4)
         (vectorized implementation for the whole image points)
5      Find the new positions of all pixel points using Equation (5) (vectorized
         implementation)
6      Find the reverse map. (See Section 5.1.3 for details)
7      Warp the sharp image using the reverse map to get warped image $W$
8      Update $average$ variable: $average = average + W$
9  end
10 $B = average/(size(\theta'))$
```

5.1.2. Blur Kernel Estimation

The blur kernels that are used as additional information to guide the image deblurring model can be jointly estimated when generating the blurred image. To describe the blur, we estimate two maps: one to describe the motion of pixels in X-axis direction and the second in Y-axis direction. The positions of image points at the start of the exposure time and the end of the exposure time are used to estimate a liner blur kernel. The same idea is used to estimate a blur kernel from the camera IMU measurements without generating a synthetic blurred image.

5.1.3. Implementation Challenges

While implementing this technique, we faced two issues. The first issue is that synthesizing the blurred image includes a nested loop. The outer loop iterates the camera poses, and the inner loop is used for calculating the homography for each pixel. The algorithm is implemented using Python. Including a nested loop with heavy computations would make the code take a long time. To speed up the code, the nested loop implementation was replaced by a carefully devised vectorized implementation.

The second issue was that the homography calculations provide the forward map for warping the image. This means that it shows how each pixel moves from its original position. However, the reverse map is needed for warping the sharp image. To achieve this, another nested loop is used to find the per-pixel warping matrix, which has the shape of the image. When finding the reverse map, the scene depth map is used to choose the pixel with the smaller depth value (*i.e.* closer to the camera). For instance, when calculating the new position of a pixel with a specific depth value, it can coincide with another pixel, with different depth, which moves to the same place. Thus, we harness the depth values to choose the foreground pixel. Including that as a nested loop inside the bigger loop would take a very long time for generating only one blurred image due to the dynamic typing used in Python language. This means that Python interpreter every time checks the variable type within the loop, which is very slow. To overcome this issue, this part of the code was implemented in C++ which follows a static typing for the variables. This was achieved using Cython library [72].

Moreover, the pixels in the image are moving differently (*i.e.* blurred differently) due to their different depth values. This causes holes (black regions) in the intermediate warped image. These holes were compensated by using an inpainting method from OpenCV library [73]. The inpainting method is the bottleneck in our code *i.e.* the most time-consuming part in our code.

5.1.4. Dataset Details

The synthetic dataset generation technique enables us to generate potentially an unlimited depth-dependent blurred dataset from RGB-D images. To enable the deep learning model to generalize well to unseen blurry images, the model should be trained with relatively huge multi-scenery data. In other words, we need a huge dataset, which spans various scenes, of sharp images with their corresponding per-pixel depth maps. Acquiring such data itself is extremely challenging and hard due to the challenges of per-pixel depth estimation, and the huge number of the needed data samples with many scene variations. For this purpose, we used the SceneNet RGB-D dataset [74]. It contains about 5 million photo-realistic, indoor synthetic RGB-D images. The Scenenet RGB-D dataset introduces 57 different layouts with

16k different configurations. The configurations are randomly chosen, so the possible configurations can virtually be unlimited.

The blur generated here is due to camera motion only (*i.e.* static scene), but it can be extended for dynamic scene situations as well.

5.2. Real Dataset Collection

In this section, we introduce a technique for acquiring blurry and sharp image pairs with their depth maps and IMU measurements.

5.2.1. The Main Idea

The main idea is to have a system setup with a camera for capturing the blurry image, a second camera for capturing the sharp image, depth estimator camera and IMU unit. The IMU unit can be embedded with the first camera capturing the blurry images (*e.g.* a modern mobile phone). The components are fixed together to capture the data. After collecting the data samples, the images from the different cameras need to be synchronized with each other. Also, the IMU measurements need to be synchronized with the start and the end of the exposure time of the first camera. The collected images from the different cameras have different views. Consequently, the images should be aligned together (*i.e.* registered).

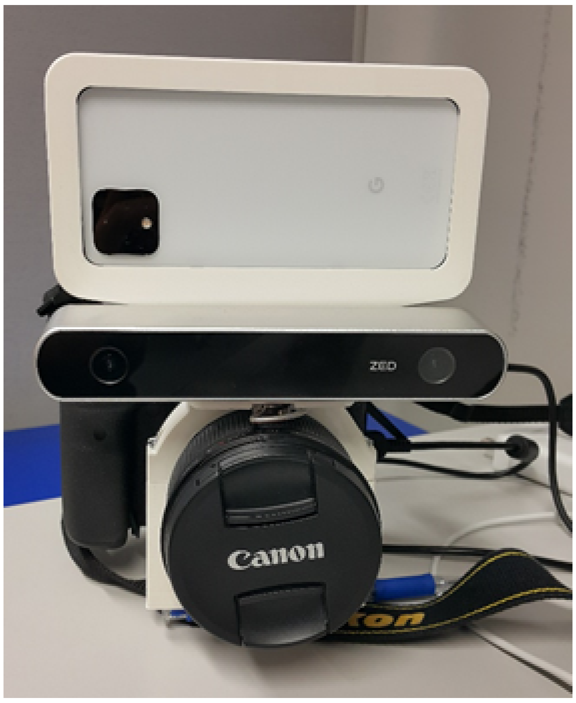


Figure 19. The multi-Camera rig. The components of the rig are as the following: Pixel4 mobile camera on the top, ZED stereo camera in the middle, and Canon DSLR camera in the bottom.

5.2.2. Multi-Camera Setup

Our multi-camera setup is a rig that contains three cameras as shown in Figure 19. The three cameras are as the following:

- Pixel4 camera: a Google Pixel4 mobile camera which is used to capture the JPEG (*i.e.* the processed RAW image) and the corresponding RAW image. In addition, the IMU measurements are recorded during the capture.
- Canon camera: a DSLR camera which is used to capture the sharp image. The camera version is Canon EOS 5D Mark IV.
- ZED camera [75]: a stereo camera which is used to capture the scene depth.

5.2.3. Data Collection Procedure

During collecting the data, we should consider the necessary information for post-processing the collected data. Therefore, the collection setup or design should include some steps that guarantee the availability of the necessary information.

The steps proposed for collecting a set of sample images are as the following:

- Take quite long videos with ZED and Canon cameras (10-20 minutes).
- Capture running millisecond-clock images with the Pixel4 camera, during the filming of the other cameras, to be used for calculating timestamp differences among the cameras.
- Capture multiple images for a checkerboard 9×6 to be used to calibrate the cameras and get the relative position among them.
- Capture other running millisecond-clock images to be used for verification.
- Capture blurry images using Pixel4 during filming by moving the camera rig while capturing.
- Capture other running millisecond-clock images, close to the end of videos, to be used for verification.

These steps are illustrated in Figure 20. The steps were devised such a way based on the problems that we faced during the preparing and testing for the cameras to collect the data.

After collecting the data, two main processing steps are performed. The first step is the data synchronization, and the second step is the alignment of the images. In the next sections, we discuss these two steps.

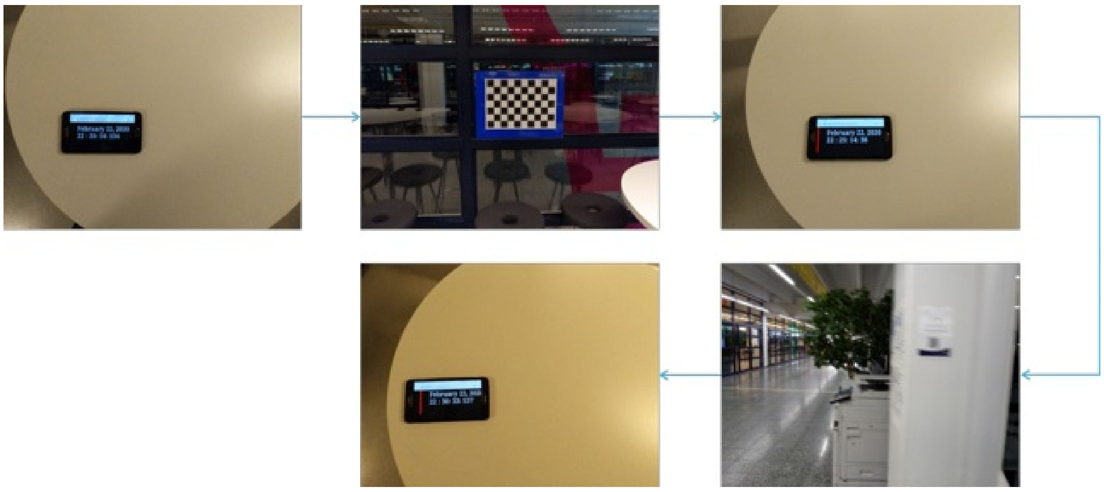


Figure 20. Data collection steps. The flow of steps is shown by the direction of the arrows. The first, third, and last images are running millisecond-clock images. The second image depicts the capturing of the checkerboard images. The fourth image is a blurry image captured by Pixel4. While capturing images with Pixel4, the other cameras, Canon and ZED are filming.

5.2.4. Data Synchronization

For the images from the different cameras to be synchronized, the capturing process should happen almost at the same time. For this purpose, a preliminary multi-threaded code was implemented for controlling the cameras and issuing the capturing command to three cameras at the same time. The most challenging camera to control was the Canon since the commands for controlling it was reverse-engineered. The multi-threaded program contains three threads for the three cameras. Each thread connects to a camera. When capturing images of a running millisecond-clock using this code, there was a difference between the captured images up to 1 second. A potential reason for such a time difference is that each camera has its own response time in addition to the delays in the communication between the computer and the cameras. The connection between the computer and both of ZED camera and Canon camera was through a USB cable, while the connection with Pixel4 was through WiFi. This huge time difference would not be suitable for image capturing while the cameras are moving since that would result in small regions of the scene shared among the cameras. Thus, controlling the three different cameras through a software is not a proper idea.

After many trials, we proposed to take long videos with the Canon and ZED camera (10-20 minutes) for each data collection session. While filming, we capture blurry images with Pixel4. At first, we capture a running millisecond-clock to be used as a reference frame. This reference frame is then picked from the three cameras to determine the timestamp of each camera in which this reference frame was captured. In this way, we have the three timestamps of the cameras as a reference for picking up the other frames. A Python code was implemented to synchronize the Pixel4 frames with other frames from the other cameras automatically. For instance, a timestamp for a Pixel4 frame is used to determine the time duration between this frame and the Pixel4 reference timestamp. This time duration is then added to the Canon reference timestamp to pickup up the nearest frame. The same steps are performed with the ZED camera. The idea behind the long video is that the Canon camera time was changing irregularly with every new video session. Moreover, it is more practical to handle a big chunk at once instead of many small chunks.

5.2.5. Data Alignment

Data Alignment, or registration, is a substantial step to have unified views from the three cameras. Each camera has its own view perspective. To map them to the view perspective of Pixel4 Camera, we un-project the 2D view to the 3D view and re-project it again to the perspective of Pixel4. Therefore, it is important to get the intrinsic matrix K, defined in (1), of each camera and the relative position between each other.

Camera Calibration. The captured chessboard (checkerboard) images are used to calibrate each camera to find its intrinsic matrix K and the distortion coefficients. Since the cameras capture the same object (checkerboard), we can determine the relative position between them. This can be implemented mainly by using two functions from OpenCV library [73], namely *"cameracalibrate"* and *"stereoCalibrate"* for individual camera calibration and finding the relative positions, respectively.

Alignment Procedure. To align the ZED view with the Pixel4 view, the 2D points are un-projected to 3D points using the inverse of the calibrated ZED camera intrinsics K_{ZED}. The depth map is used to rescale those 3D points. The 3D point cloud is then transformed to the Pixel4 camera coordinates using the calibrated relative rotation and translation of Pixel4 camera to ZED camera. The 3D point in Pixel4 camera coordinate frame is then projected to the Pixel4 image coordinates using the calibrated Pixel4 camera intrinsics K_{P4}. Warping the ZED image is only needed for verification of the quality of the alignment. Similarly, the depth values are warped to Pixel4 perspective.

The 3D points in the ZED camera coordinate frame are transformed to the Canon camera coordinate frame using the relative rotation and translation between ZED and Canon cameras, and then to the Canon image coordinates using the Canon calibrated camera intrinsics K_{can} to get the 2D points. Since the 3D points mapped to the 2D Pixel4 image coordinates and the 2D Canon image coordinates, we have a 2D-to-2D mapping between the Canon and the Pixel4 which is then used to warp the Canon view to Pixel4 view. Also, we can think of this process as we map the 2D Canon texture (Pixels values) to the 3D cloud points which are mapped to the projected 2D image points of Pixel4.

To show the calculation flow of the un-projection and re-projection, let us suppose we have camera1 and camera2. We need the image of camera1 to be warped to the view perspective of camera2.

To unproject a 2D point $\widetilde{x}$ in homogeneous image coordinates of camera1, which is represented by a 3×1 vector:

$$X = K_1^{-1}\widetilde{x} \tag{19}$$

where K_1 is camera1 intrinsic matrix, and X is the up-to-scale 3D point. The 3D point X is divided by its third element $X^{(2)}$ (assuming zero-based indexing, and the superscript brackets notation is used to refer to a vector indexing) and multiplied by the corresponding depth value as follows:

$$X_z = X * z / X^{(2)} \tag{20}$$

where X_z is the 3D point in camera1 coordinate frame, and z is the corresponding depth value of the point. The 3D point $\widetilde{X_z}$ in the homogeneous coordinate frame of camera1 is transformed to camera2 coordinate frame as follows:

$$\widetilde{G} = \left[\begin{smallmatrix} R & T \\ 0 & 1 \end{smallmatrix}\right] \widetilde{X_z} \tag{21}$$

where R,T are the 3×3 rotation matrix and the 3×1 translation vector, respectively. They represent the relative camera2 pose to camera1, and form the 4×4 camera2 extrinsic matrix. $\widetilde{G}$ is a 4×1 vector which represent the 3D point in the homogeneous coordinate frame of camera2. We divide $\widetilde{G}$ by the scale element $\widetilde{G^{(3)}}$ such that $\widetilde{G'} = \widetilde{G}/\widetilde{G^{(3)}}$, where $\widetilde{G'}$ is the 3D point in homogeneous coordinates of camera2 with the scale element equal to 1. The 3D point G', which is in non-homogeneous coordinate, is then projected to the image coordinate of camera2 as follows:

$$\widetilde{g'} = K_2 * G' \tag{22}$$

where K_2 is the camera2 intrinsic matrix, and $\widetilde{g'}$ is the scaled 2D point which is divided by the scale element $\widetilde{g'^{(2)}}$, and transformed to a non-homogeneous representation to get the 2D point g in image coordinates of camera2.

Alignment Challenges. The process of alignment is not straightforward and many issues result in undesired artefacts that are not trivial to be eliminated. The following issues affect on the success of the image alignment: occlusions, calibration errors, depth map errors, temporal shift, and rolling shutter effect. We explain each point in the next paragraphs.

Occlusions. The three cameras (centres of the cameras) naturally have different positions in the rig. Therefore, a camera can reveal a background object details which are occluded by another camera. When warping the Canon or ZED view to the perspective of Pixel4, the non-occluded details in Pixel4 images are interpolated from the occluding objects in ZED or Canon image, or the neighbouring pixels in these images. Making the camera as close to each other as possible can help to relieve this issue, but it is inevitable.

Calibration Errors. As shown in Equations (19,20,21,22), the calibration parameters: the intrinsic camera matrices, the rotation, and translation vectors are used for un-projection and projection. Therefore, the errors in those calibration parameters can produce errors in the alignment process.

Depth Map Errors. The scene depth is retrieved using the ZED camera. There might be inaccurate depth values that can lead to an incorrect projection of the 3D points to the other views. Furthermore, some regions are undefined due to the limited depth range of the camera, which is limited by the baseline of the stereo camera, or the stereo matching difficulties which can result from the surfaces with a regular texture or reflections.

Temporal Shift. As we discussed in Section 5.2.4, the synchronization process of a Pixel4 image with the frames of the other cameras is performed by picking up the frame with the nearest timestamp from ZED or Canon camera. Thus, there might be a slight temporal shift between the Pixel4 image and the frame of ZED or Canon cameras. This slight temporal shift between the images does not affect the alignment process when the cameras are steady. However, since the cameras are moving during the capturing process, the temporal shift can cause virtual shifts in the calibrated positions of the cameras (*i.e.* their real positions) as illustrated in Figure 21. In other words, this makes the cameras as if they have different relative positions from what they are in the real camera rig setup. The effect of the temporal shift appears as a shift between the aligned views as illustrated in Figure 22. The amount of the temporal shift depends on the video frame rate and the movement of the cameras.

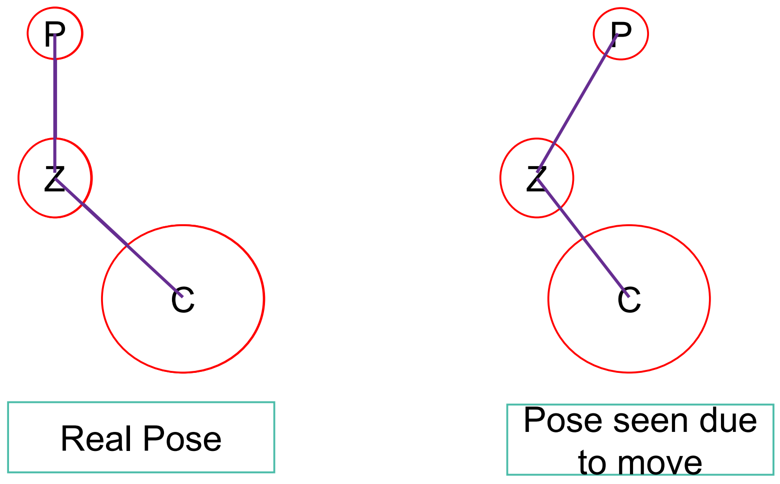


Figure 21. Temporal shift illustration example. C, Z, and P represent the Canon, ZED, and Pixel4 cameras, respectively. The cameras appear as if they are shifted from their real positions within the rig.

As a consequence of the temporal shift, the offline calibration we perform at the beginning is not sufficient, and other processing steps are required to handle the shifts between the aligned images. To relieve the temporal shift effect, we propose an online image-based calibration technique. At first, we detect SIFT [76] features between the ZED image and Pixel4 image. The 3D point cloud is already retrieved from the ZED image and scene depth as explained at the beginning of this section. The 3D cloud points are in the ZED camera coordinate frame. Thus, By having the 2D-to-2D SIFT features matches and their corresponding 3D points, the problem turns into a problem of finding the best pose of the Pixel4 camera that projects the 3D points to these 2D points in the Pixel4 image *i.e.* finding the best relative pose of Pixel camera to the ZED camera as illustrated in Figure 22. The same procedure is then applied to the Canon camera, and for every image in the dataset. For finding the camera pose, we use the *solvePnP* method with several variations of its parameters. Besides, we use *solvePnP* with RANSAC scheme [77] to be resistant to outliers. The best alignment is then chosen by comparing the image and the warped image together.

Rolling Shutter. A camera can be equipped with a rolling shutter or global shutter. In contrast to the global shutter cameras, the rows of pixels in the rolling shutter camera

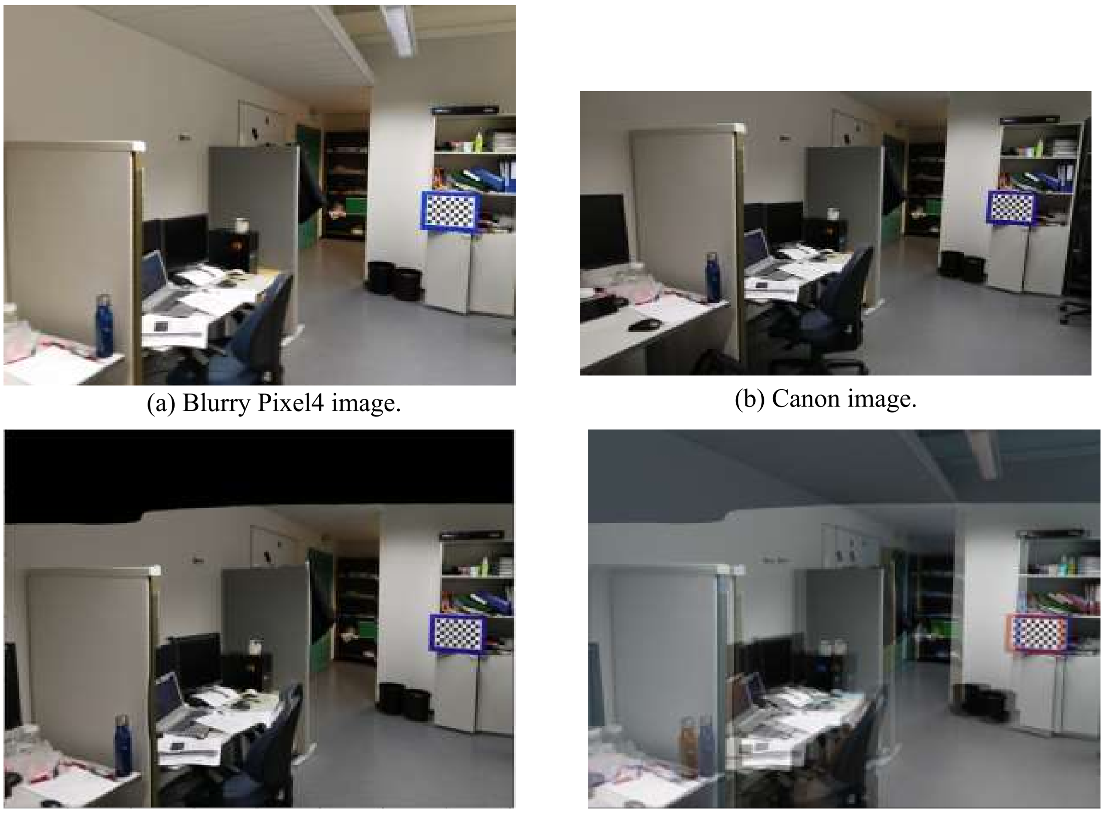


Figure 22. The effect of the temporal shift on the alignment process. (c) shows the warping result of the Canon image in (b) to the perspective of Pixel4 image in (a). In (d), the Pixel4 image (with red checkerboard) overlays the warped Canon image to visualize the temporal shift.

are not read at the same time, but they are slightly read at different times. If the camera is steady, the rolling shutter does not affect the captured image. However, if the camera is moving, the rolling shutter can cause tilting artefacts. For instance, if the camera is rotating around its Y-axis, the straight vertical edges in the scene can appear tilted. Consequently, this can affect the alignment process in general. Moreover, the rolling shutter can especially hinder our proposed method for relieving the temporal shift. The method for finding the best camera pose (*solvePnP*) might produce higher reprojection error because not all detected points satisfy the found camera model. As a result, if the *solvePnP* method is used with RANSAC, this may decrease the number of points used for finding the camera pose which, in turn, produces errors in the alignment of the images due to the non-sufficient number of points used for finding the camera pose.

5.2.6. Data Masks

Finding the mask that cover all the artefacts produced by warping images to the perspective of Pixel4 is challenging. The simple pixel-by-pixel comparison between the Pixel4 image and the warped image is not preferable since the Pixel4 image is

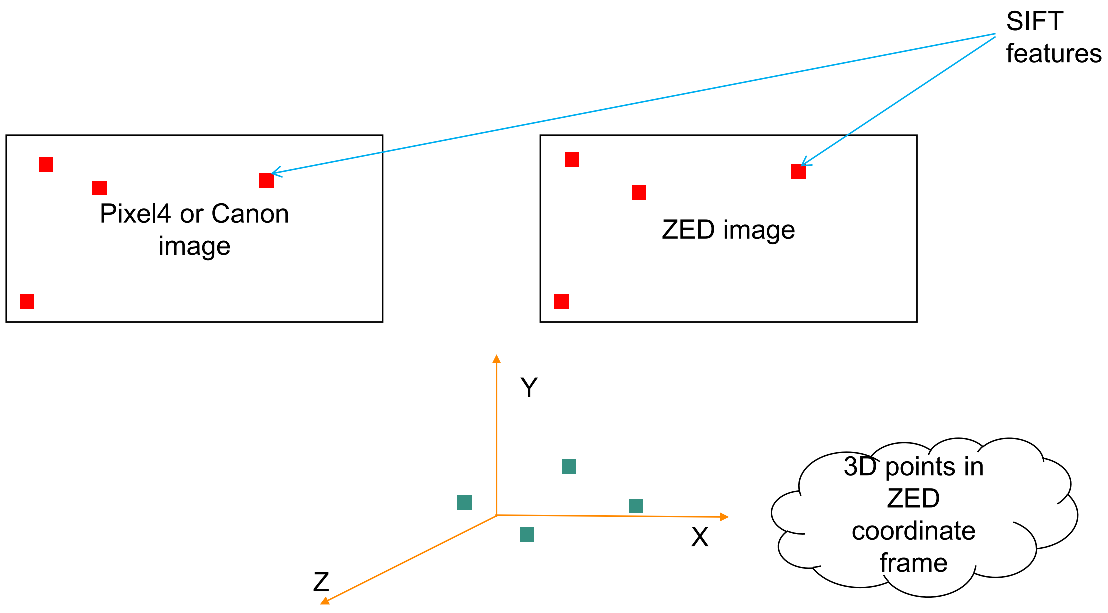


Figure 23. An illustration of the online image-based technique which is followed to relieve the temporal shift effect.

blurry. Thus, the goal of the mask is to cover most of the artefacts in the warped image.

To cover the artefacts of the occlusions, the details appearing in the ZED image and occluded in the Canon image are detected by exploiting the scene depth map (or the 3D point cloud). This is performed by tracking the projection of every 3D point to the perspective of the Canon camera. If at least two points are projected to the same 2D pixel in the image, this means that one of those points is occluded by the other point. The depth map (the Z value of the 3D point) is used to decide which point is in the foreground. The background point is then marked in the matrix of the 3D point cloud as occluded. After tracking the whole 3D points, the marked points form a part of the final mask. The masked area is increased by morphological operations. This will help to cover the occluded details in the Canon image and appearing in the Pixel4 image due to the positions of the cameras in the rig. The same steps are followed with the Pixel4 image and ZED image to detect the details appearing in the ZED image and occluded in the Pixel4 image.

Since the warped Canon image and the ZED image are both sharp, we can perform a pixel-by-pixel comparison between them. This helps to cover the artefacts produced by the reprojection or the shifts. Besides, it helps to cover the different reflections from the surfaces.

The undefined or infinity depth values are marked in the mask as well. The final mask is then the sum of all those masks.

5.3. Network Architecture

Our network architecture is a fully CNN that has the U-Net shape discussed in Section 4.2.8. The additional information layers are stacked with the blurry image. The corresponding convolution layers in the encoder and decoder are concatenated with skip connections. The blurry image input is added to the output of the last convolution layer with a residual connection. This residual connection has been shown to introduce an improvement in the deblurred images when used. The residual connection can enable the network to learn the residual between the deblurred image and the blurry image, which is easier for the network. In addition, this can enable the network to maintain colour consistency between the input and the output [49].

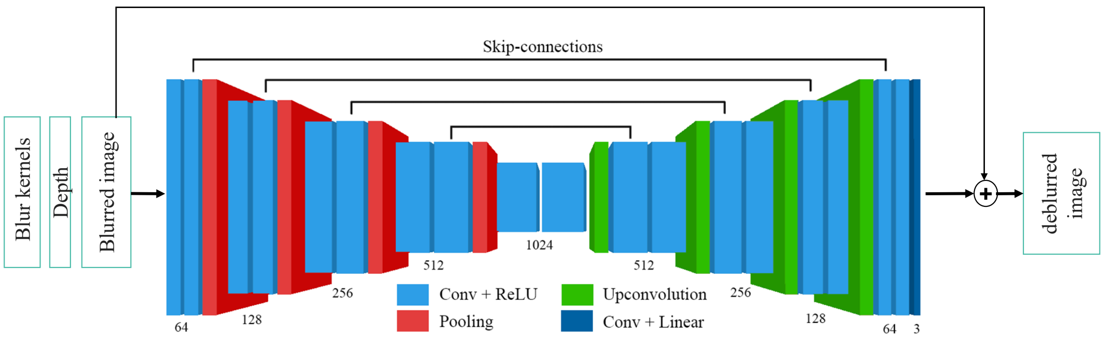


Figure 24. Our network architecture. The inputs to the network are stacked. All the convolution filters are of size 3×3, except the last layer filter which is 1×1. The size of upconvolution filters are 2×2. The pooling layer performs 2×2 max pooling. The skip connections concatenate the corresponding layers between the encoder and decoder. In the top, a residual connection is used between the blurry image input and the output of the last convolution layer. This network is a modified version of [50].

6. EXPERIMENTAL RESULTS

We introduce the results of training the network with our generated synthetic dataset. We show the results of providing additional information to the network to guide the image deblurring process. The additional pieces of information are the estimated blur kernel and the depth map. Our results are compared with the state-of-the-art deblurring method DAVANet [49]. In addition to the generated synthetic data, real blurry and sharp image pairs are collected. The real data is used to train the network from scratch, or finetune the model pre-trained with the synthetic dataset. To show the effectiveness of the additional information on the image deblurring process, we train the network in the following variations: no-additional information (*i.e.* using only the blurred image), providing depth information only, estimated blur kernels only, or both of them.

6.1. Evaluation Metrics

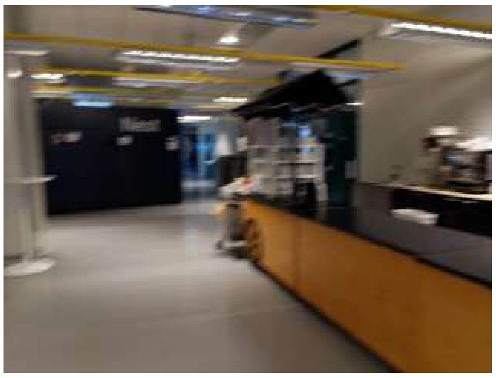
There are many evaluation metrics to compare the results of the deblurred image against the reference latent sharp image. The most used evaluation metrics are the peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) [78]. In addition, several evaluation metrics were proposed when the reference sharp image does not exist [27]. Moreover, in [27], they show that VIF [79] metric is more effective in case of non-uniform deblurring. In our results and comparisons, we will utilize SSIM and PSNR metrics.

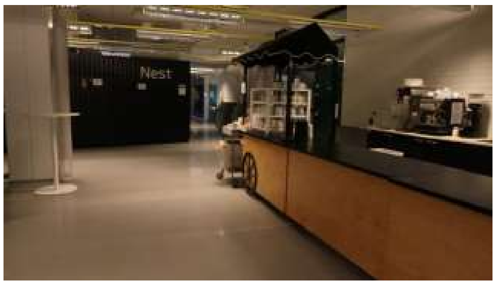
6.2. Synthetic Dataset Details

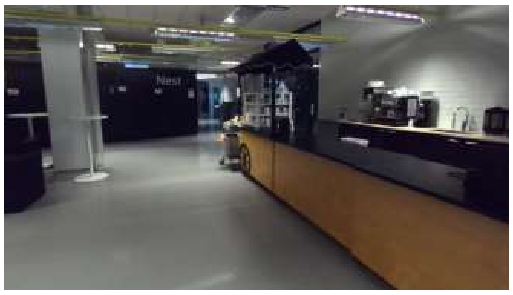
We use SceneNet RGB-D [74] (5 millions of synthetic images) dataset to generate depth-dependant blurred images for deep neural network training. The generated dataset consists of 27K spatially-variant blurred and sharp image pairs with their corresponding estimated blur kernels and depth maps. The 27K-dataset was generated using 9K different configurations. The dataset was split to a training set with 24k samples (8K different configurations), and testing set with 3k samples (1K different configurations). The training dataset generation was computationally performed across four separate processors to speed up the generation process. Random noise from Poisson distribution is added to the synthetically generated blurred image. Alternatively, random noise can be added during the training phase of the network, however, this would slow down the training process a little bit. The noise addition is shown to be important to regularize the network and enable it to generalize well to the unseen data and avoid overfitting the training dataset. The resolution of the used images is 320×240 pixels.

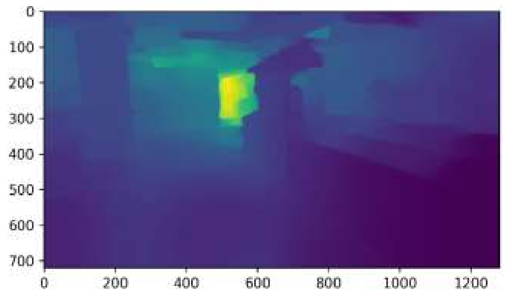
6.3. Real Dataset Details

Following the procedure explained in Section 5.2.3, we collect a real dataset of blurry and sharp image pairs with their corresponding IMU measurements and scene


(a) Blurry image.


(b) Canon image.


(c) Right ZED image.



(d) ZED scene depth.

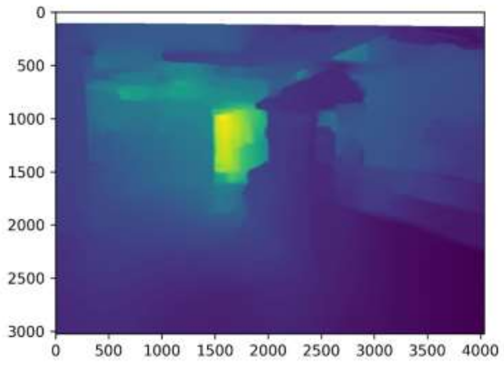
Figure 25. A real data sample where (a) is the Pixel4 image, (b) is the corresponding canon frame, (c) is the corresponding ZED right frame, and (d) is the ZED depth scene. Warping of the corresponding information to the Pixel4 is shown in Figure 26.

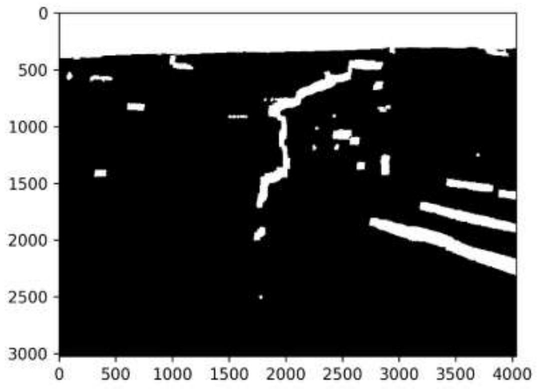
depth maps. After collecting the data, the images from the different cameras are synchronized and aligned, and the image mask is generated as discussed in Section 5.2. The real dataset consists of 1649 image collected from 5 different scenes. The dataset is split to 1273 samples (3 scenes) for training, and 376 samples (2 scenes) for testing.

A sample image from the collected dataset is shown in Figure 25. The sample contains a blurry image and its corresponding Canon image, Zed right image and ZED right depth map. This data sample shows how the three cameras are synchronized. The alignment of the Canon image and ZED depth of this example is shown in Figure 26. Also, the generated mask that covers the incorrect pixels or regions is shown in Figure 26. To visually verify the quality of the warped Canon image (*i.e.* the sharp image) and the image mask, the mask and the warped image are overlaid on the original blurry image from Pixel4, as illustrated in Figure 26.

The resolution of Pixel4, ZED, and Canon images are 4032×3024, 1280×720, and 1920×1080, respectively.

(a) Sharp image - warped from Canon image.


(b) Scene depth map - warped form ZED depth.


(c) Image mask.

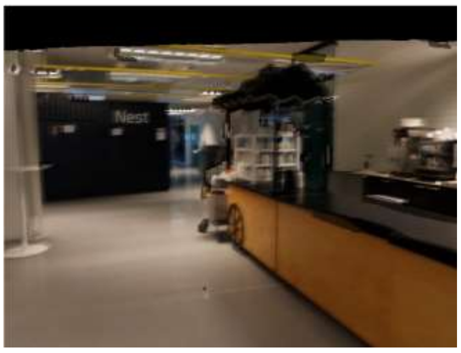

(d) Overlay mask and warped canon over Pixel 4 image.

Figure 26. Warping of Canon image and ZED depth to the perspective of Pixel4. In (d), the mask and the warped Canon image are overlaid on the original blurry Pixel4 image for visual verification of the quality of the alignment.

6.4. Network Training

In this section, we discuss the details and results of training the proposed network using both synthetic and real datasets.

6.4.1. Using Synthetic Dataset

The network is trained for 100 epochs. The used batch size is 32 samples. With the 24K-dataset for training, this means that the network performs 75K iterations. We use Adam optimizer [34] with parameters $\beta_1 = 0{,}9$ and $\beta_2 = 0{,}999$. The initial training rate is set to 10^{-4} and decayed by $0{,}1$ after the 80^{th} epoch. We use the Keras platform for our implementation.

In our experiments, we perform the following variants of network training:

- The blurred image only is provided as input to the network. In this context, we use the word *"blind"* to indicate that no additional information is provided with the blurred image.

- The depth map is provided with the blurred image as an input. The depth map values are rescaled between 0-255.
- The depth map and the blur kernels are provided with the input.
- Both of the depth map and blur kernels are provided with the input.

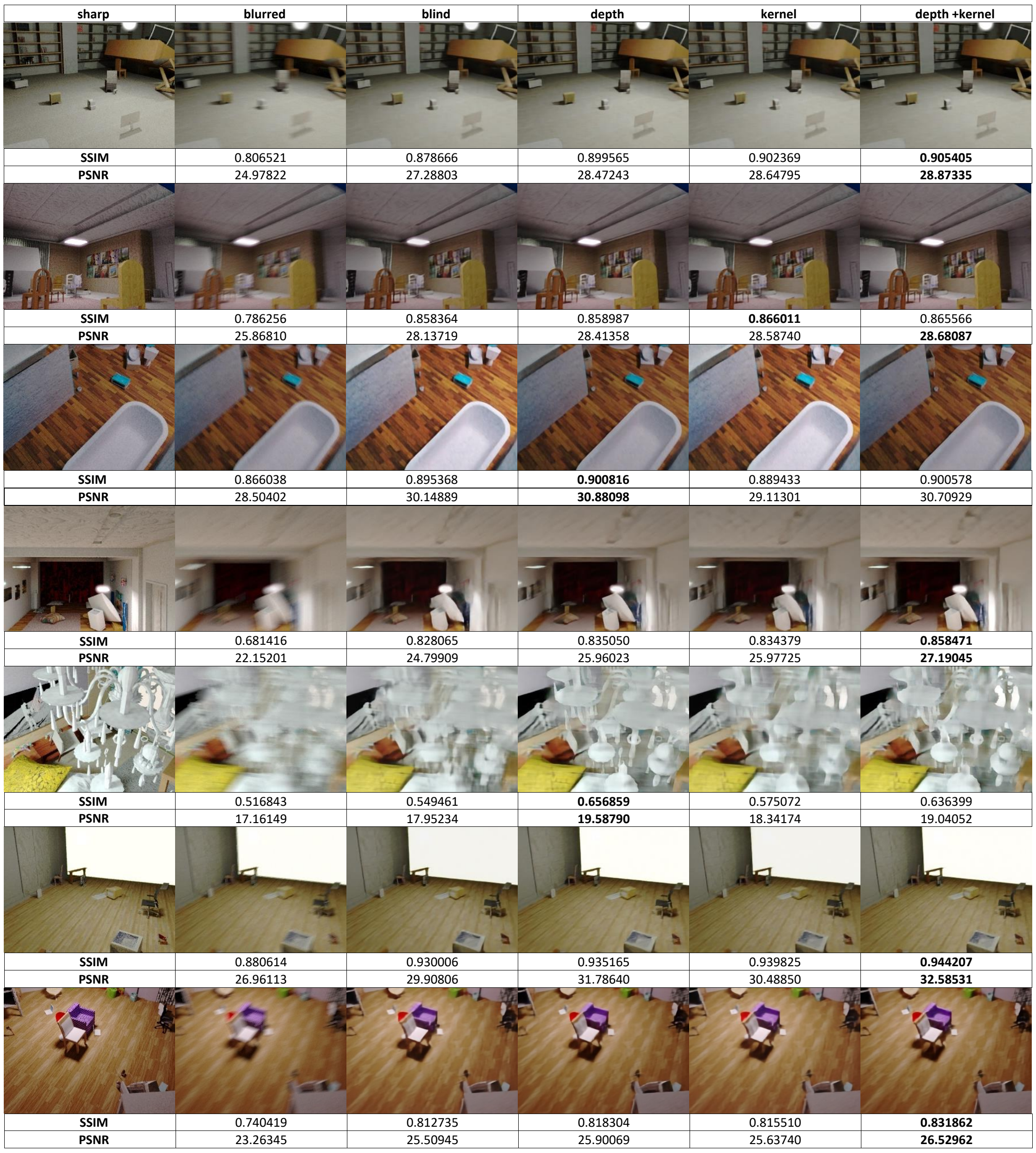
In these experiments, the additional information is stacked as additional input layer with the blurred image. We refer to our neural network trained with our synthetic dataset as the *synthetic model*. Besides, we compare our results with the state-of-the-art deblurring method DAVANet [49]. We refer to the DAVANet model that deblur a single blurry image without any additional information as *single DAVANet*, and the model, which deblur a stereo image exploiting the depth information and view aggregation as *stereo DAVANet*. Also, we train their single DAVANet with our synthetic dataset, and refer to this model as *synthetic DAVANet*.

The result of training our model with the synthetic dataset is shown in Figure 27. In all cases, leveraging of either the estimated blur kernels or the scene depth map shows better results in comparison with the blind case. Furthermore, in most cases, exploiting of both pieces of information enable the network to perform better in terms of quality of the deblurred image. In the fifth row of Figure 27, an interesting example shows how the scene depth helps to recover much more details in the chandelier which is very close to the camera and suffers from more blur in contrast to the other objects in the scene.

To compare our results with single DAVANet, we train their single model with our synthetic dataset. In Figure 28, the results show that our network performs better when exploiting the scene depth map or the estimated blur kernel. Moreover, in terms of SSIM, our synthetic model outperforms the single DAVANet model in all cases when leveraging the scene depth and the estimated blur kernels.

In addition to the synthetic test images, we further test our model with real blurry images, and compare the results with synthetic DAVANet, single DAVANet, and stereo DAVANet. In Figure 29, the outputs of our network show competitive results with single DAVANet. On the other hand, the results show improvements in the deblurred images of synthetic DAVANet which is trained with our synthetic dataset. This shows how our dataset is diverse, and enables the model to generalize much better than the DAVANet proposed dataset. The deblurring of the single real blurry images using stereo DAVANet produced heavy artefacts on the outputs. In DAVANet [49], their work includes testing the stereo DAVANet with a one-sided image from their dataset which does not produce any obvious artefacts.

Moreover, we captured real blurry images using ZED stereo camera. In Figure 30, our synthetic dataset showed clearly better results when used to train the DAVANet network in comparison to single and stereo DAVANet. This assures the effectiveness of our dataset when utilized for training the DNN models. The DAVANet originally proposed utilizing stereo images for deblurring. However, the results of deblurring real blurry stereo images are almost still blurry with heavy artefacts. In their work, they did not include any results of deblurring real blurry images.

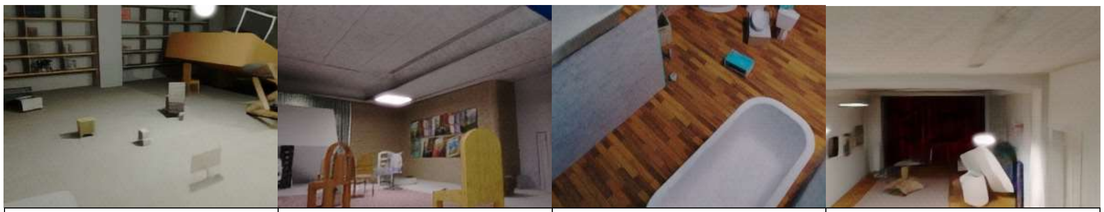


	sharp	blurred	blind	depth	kernel	depth +kernel
SSIM		0.806521	0.878666	0.899565	0.902369	**0.905405**
PSNR		24.97822	27.28803	28.47243	28.64795	**28.87335**
SSIM		0.786256	0.858364	0.858987	**0.866011**	0.865566
PSNR		25.86810	28.13719	28.41358	28.58740	**28.68087**
SSIM		0.866038	0.895368	**0.900816**	0.889433	0.900578
PSNR		28.50402	30.14889	**30.88098**	29.11301	30.70929
SSIM		0.681416	0.828065	0.835050	0.834379	**0.858471**
PSNR		22.15201	24.79909	25.96023	25.97725	**27.19045**
SSIM		0.516843	0.549461	**0.656859**	0.575072	0.636399
PSNR		17.16149	17.95234	**19.58790**	18.34174	19.04052
SSIM		0.880614	0.930006	0.935165	0.939825	**0.944207**
PSNR		26.96113	29.90806	31.78640	30.48850	**32.58531**
SSIM		0.740419	0.812735	0.818304	0.815510	**0.831862**
PSNR		23.26345	25.50945	25.90069	25.63740	**26.52962**

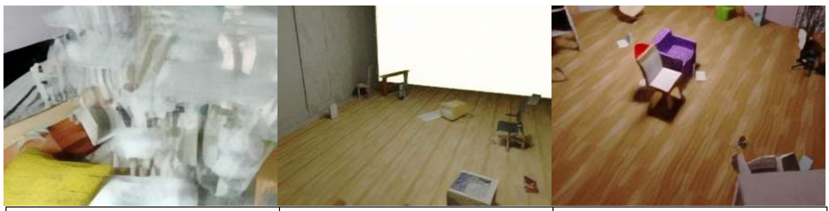
Figure 27. The image deblurring results of our synthetic model in the following cases: blind case, using the scene depth information, using the estimated blur kernels, and using both the scene depth and the estimated blur kernels. The image deblurring results are shown in terms of SSIM and PSNR. A bold-font value indicates the maximum value among the other cases in the same row.

6.4.2. Using Real Dataset

The pretrained model with the synthetic dataset is finetuned using the collected real dataset. As a rule of thumb, and since this dataset is much smaller than the synthetic dataset, we set the learning rate to 10 times smaller value than what we used in training with the synthetic dataset. The finetuning is performed for 15 epochs with a batch size of 2 samples. The whole model parameters are finetuned *i.e.* all model layers are finetuned.



0.897543	0.855616	0.894088	0.821838
29.01825	28.47731	30.51254	25.56508



0.545727	0.936519	0.813409
18.06021	31.29509	25.65826

Figure 28. The results of deblurring the blurred images in Figure 27 by the single DAVANet model trained with our synthetic dataset. The synthetic DAVANet results are shown in terms of SSIM in the first row, and PSNR in the second row. A bold-font value indicates the maximum value among the other cases in the same row of Figure 27.

When deblurring samples from the real collected dataset, the samples seem to be almost blurry with only very slight restoration cues. This happens because our synthetic model experiences a very small number of samples with large blur sizes. The resolution of the Pixel4 images is over 100 times larger than the synthetic images. Thus, the blur kernels are vastly larger than what the synthetic blurred images have. To adapt our network for the large blur kernels, we randomized the parameters of the intrinsic camera matrix to account for the different blur kernel sizes.

At first, some samples of the real images are deblurred using the pretrained synthetic model to act as a baseline for the results of finetuning the model. In other words, the results of finetuning are compared to the results of the pretrained model to observe how the finetuning affects the performance of the model in terms of the quality of deblurring real images. Moreover, Figure 31 shows that our synthetic model deblurring results outperformed the single DAVANet model results. The single DAVANet shows no significant improvement on the deblurred real images. Also, we show preliminary results of deblurring the real image with the help of the scene depth map. The depth map is used without applying the data mask to remove the incorrect regions resulted from warping the ZED depth map. Thus, it is expected to experience some artefact when deblurring the image using the depth map. However, the goal of this experiment is to observe the effectiveness of the scene depth on the deblurring process. Although the deblurring results with the depth map slightly changed, one can still observe a slight improvement on the foreground object in the first-column image in Figure 31.

The results of deblurring the real images after finetuning the pretrained synthetic model are shown in the last row of Figure 31. The results suffer from heavy blur-like artefacts. The reason behind these artefacts is that the real blurry and sharp image pair is not perfectly aligned. To further figure out how we can relieve this problem, we finetuned the model with a resized version of the real images, which is the same as

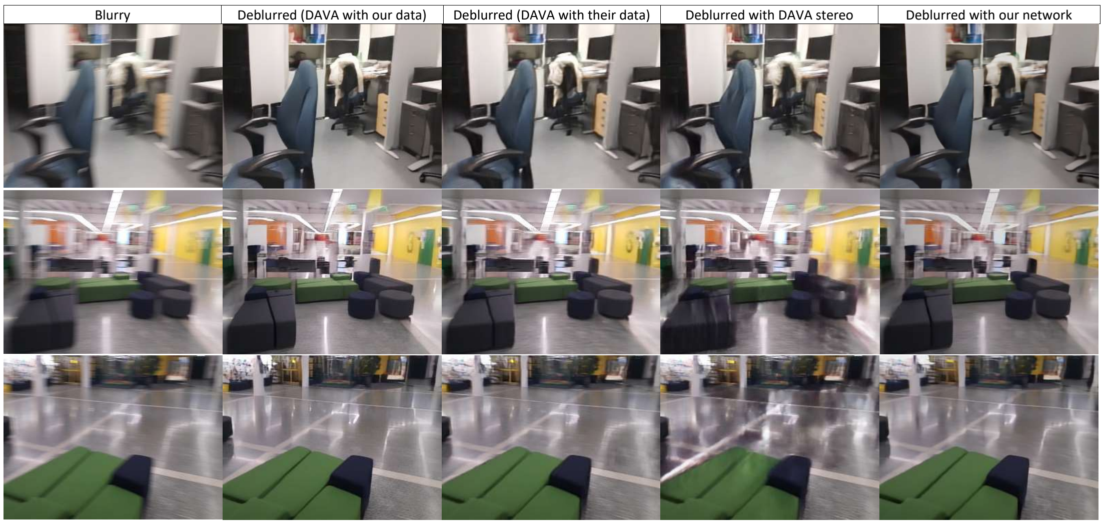

Blurry
Deblurred (DAVA with our data)
Deblurred (DAVA with their data)
Deblurred with DAVA stereo
Deblurred with our network

Figure 29. The deblurring results of real blurry images with the following models: synthetic DAVANet, single DAVANet, stereo DAVANet, and our synthetic model, respectively. Utilizing our synthetic dataset to train DAVANet model improved the visual results of the outputs.

the resolution of the synthetic blurred dataset. However, the problem still persists. In addition, when we finetuned the model with a chosen subset of DAVANet subset, the blur-like artefacts do not appear. Nevertheless, we noticed a decrease in the quality of the deblurred synthetic images. Therefore, even if the data is well aligned, the finetuning process should be handled very carefully as well. It becomes clear that the misalignments need further handling.

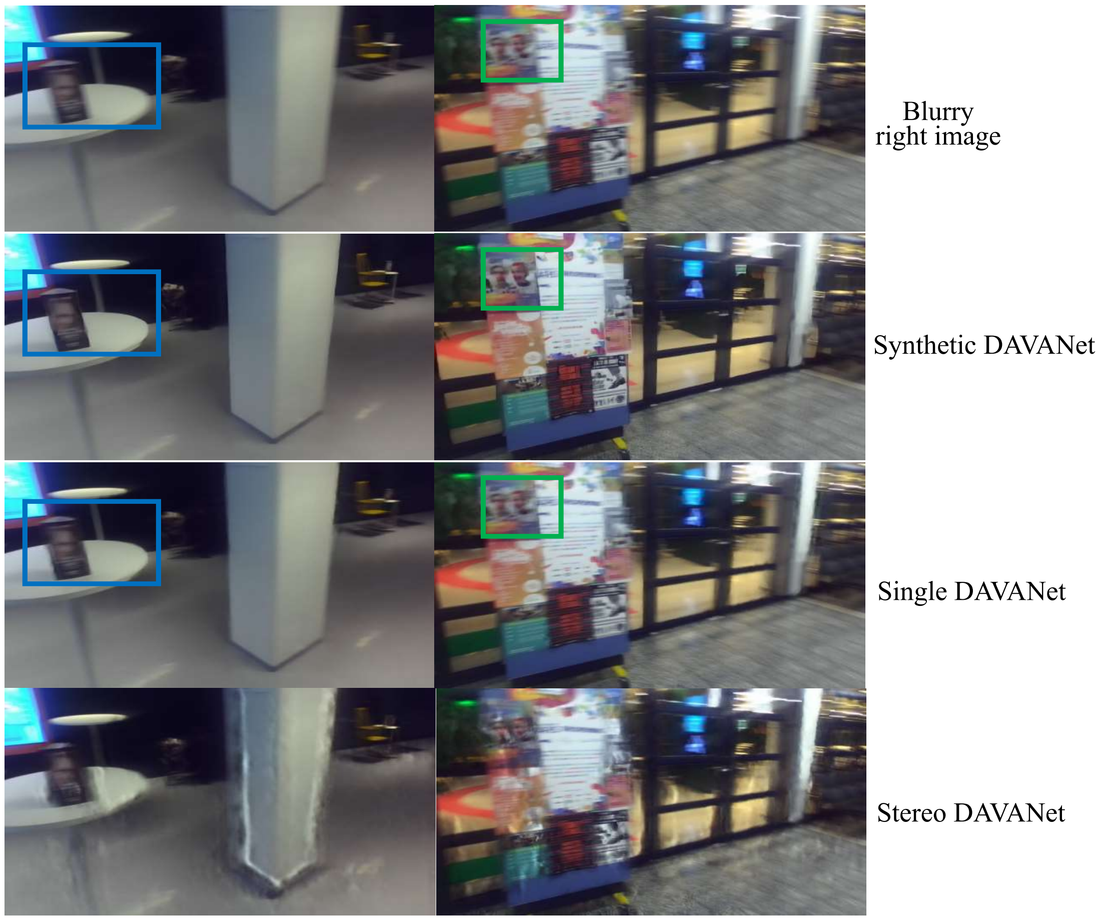


Figure 30. Deblurring results of real blurry stereo images (only one side is shown here) by the following models: synthetic, single, and stereo DAVANet. The results of synthetic DAVANet is visually much better than single DAVANet as shown in the green and blue boxes. The stereo DAVANet seems to be blurry with heavy artefacts.

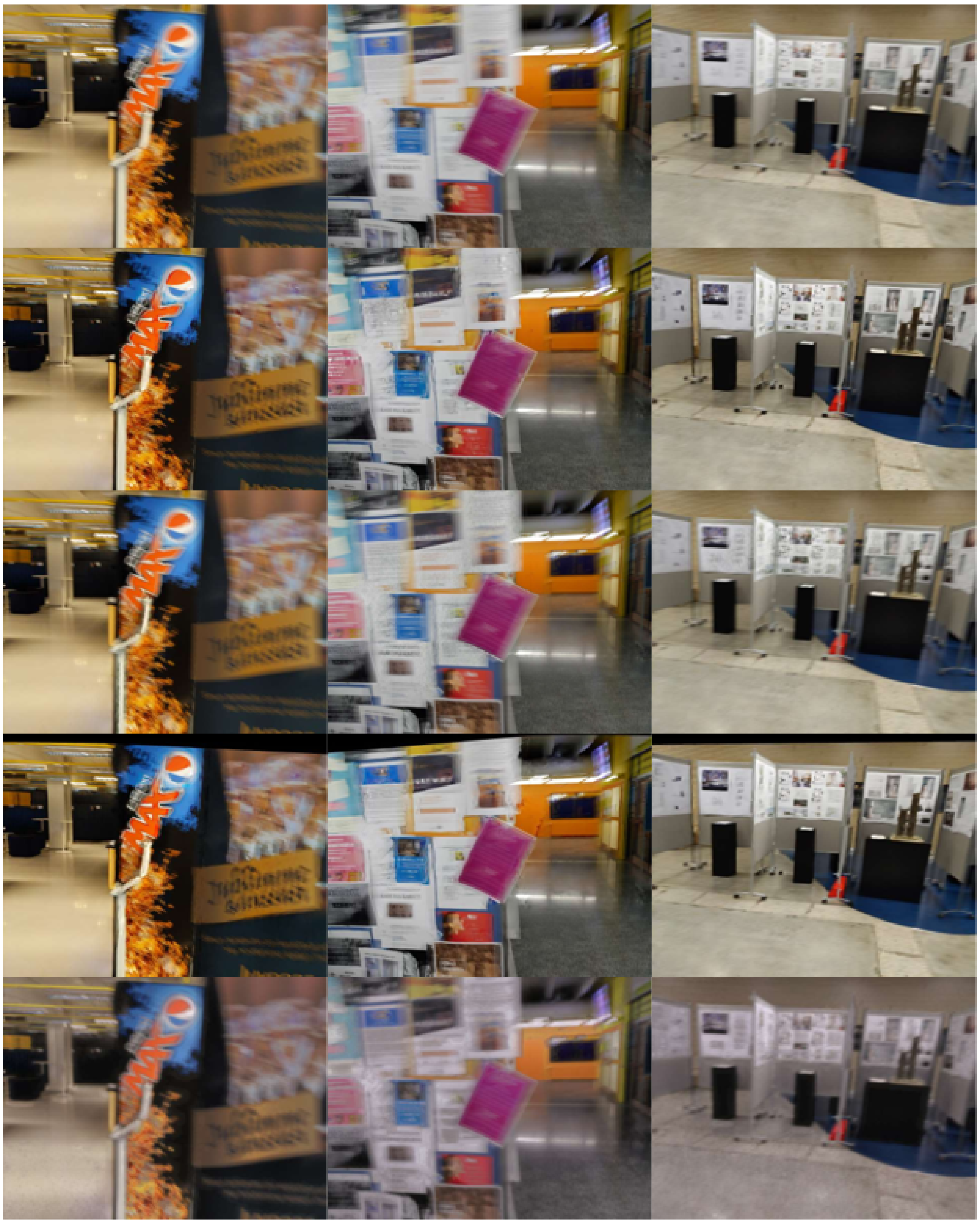


Figure 31. The results of finetuning the pre-trained synthetic model. The figure shows a visual comparison of deblurring the blurry images of the first row using the following models (from the top to bottom): blind synthetic model, single DAVANet, synthetic model with the depth map, and the fine-tuned model. The black regions in the images, deblurred using the synthetic model with the depth map, indicate undefined depth values.

7. DISCUSSION

In this thesis, we proposed a deep learning-based approach for image deblurring which is guided by the scene depth information and IMU measurements. The conventional techniques, discussed in Chapter 3, are generally non-robust against non-uniform blur. In addition, they are computationally time-consuming. However, many ideas from conventional techniques can be beneficial for learning-based approaches. For instance, leveraging of the scene depth and camera's IMU has been studied in several conventional methods [13, 5, 15, 16].

The network architecture and the availability of huge training dataset are the most challenging aspects of adopting a DNN-based image deblurring solution.

In the context of providing training data, we proposed a novel technique which can potentially synthesize unlimited spatially-variant blurred images. The scene depth variations and 6D camera motions were considered when synthesizing spatially-variant blur. The SceneNet RGB-D dataset [74], which contains 5 million photo-realistic indoor images, was used to generate a large amount of blurred images. Although it consists of 57 different layouts only, the different configurations that can be drawn out from these layouts are virtually unlimited. The number of configurations in their dataset is more than 16k. Our synthetic 24k-dataset has been shown to be effective in blind deblurring (*i.e.* with no additional information) against the single and stereo DAVANet [49] model trained with their dataset. The DAVANet dataset contains 20,637 blurry and sharp image pairs from 135 diverse scenes. Although this number is over two times larger than the number of layouts in SceneNet RGB-D dataset, the enormous number of configurations produced from these layouts played a vital role in diversifying the dataset. Moreover, training the single DAVANet model with our synthetic dataset clearly improved the prediction abilities of the model, and enabled it to generalize well to unseen real blurry data.

Training the single DAVANet with our synthetic dataset enabled the model to perform better than our blind model. The model architecture differences played a vital role. There were two essential differences between the single DAVANet and our model. The first was the usage of the residual blocks as the core building block of the DNN model instead of cascading usual convolution and pooling layer outputs. The second difference was the concatenation of a number of differently dilated convolution layers (*i.e.* the context module) to increase the receptive field for the next layers without increasing the model parameters. These two differences seem to make the model able to learn better. While the usage of the context module was supposed to embed multi-scale features, the single DAVANet was not able to handle different blur kernel sizes. For future development, it is desirable to devise model architectures that can handle images blurred with significant variations of blur sizes. However, considering the total number of parameters in the model is crucial for computation-wise performance. Besides, it has been shown that utilizing a residual connection between the blurry image input and the output can slightly improve the result. Understanding the ideas behind such components and the convolutional networks generally can tremendously help to design better model architectures for addressing the challenging problem of the image deblurring. Moreover, this should be combined with proper modelling of the blur and studying of its properties.

In our work, we showed that leveraging both the scene depth and the camera's IMU measurements produce better results for deblurring spatially-variant blurred images. These results can be further extended to utilize sparse or noisy depth maps instead of dense or per-pixel maps. In our work, the pieces of additional information are stacked as additional input layers with the blurry image. However, more sophisticated approaches can be considered for fusing the additional information with blurry image information for better utilization of the aggregated information. Additionally, this can help the network to handle the sparse or noisy information in both depth map and IMU measurements. To observe the effect of utilizing scene depth, it was substantial to generate blurred images which have spatially-variant blur kernels due to arbitrary camera motions and scene depth variations. This was achieved by our novel synthetic dataset generation technique. The blur kernels estimated from the IMU measurements are linear. However, it is possible to consider non-linear blur kernels and propose alternative ways to feed this information to the neural network.

Our solution is proposed for handling blur in static scenes. However, our synthetic dataset generation tool can be extended to consider deblurring in dynamic scenes. For future directions, one can consider leveraging of multiple frames to exploit the various view information in the different frames. Furthermore, the utilization of multi-frame input can be beneficial in case of dynamic scenes to estimate the motion of the objects from the multiple frames.

In addition to our effective synthetic dataset generation tool, we proposed a promising multi-camera setup for collecting real blurry and sharp image pairs and their corresponding scene depth maps and camera's IMU measurements. The most challenging aspect was the alignment of the view from the different cameras, which was hindered by several issues as discussed in Section 5.2.5. While the collection of real blurry and sharp image pairs is extremely challenging, we have passed a long track of development in that direction and introduced promising results for future continuation of addressing the challenges of image registration. On the other hand, the multi-camera system can be changed and further developed to relieve the problems of synchronization and alignment of images. For instance, a hardware solution can embed three homogeneous cameras in one device, where two cameras with a wide baseline are used for estimating the depth and the third camera is very close to one of them.

Despite the misalignment in the collected real data, we fine-tuned our synthetic model to observe how the results of image deblurring improve when using the real blurry and sharp images. However, blur-like artefacts appeared in the outputs of the deblurring model. Thus, it became clear that the misalignment between the blurry and reference sharp image requires further handling. In future, we can also consider embedding a motion flow solution in our neural network model to deal with the misalignment in the blurry and sharp image. We also tried to finetune the model from a carefully chosen data sample form the DAVANet dataset [49]. While the blur-like artefact did not appear, the performance of the synthetic image deblurring decreased a little bit. This indicates that the process of finetuning should be approached carefully as well.

8. CONCLUSION

In this thesis, the scene depth and the IMU measurements were utilized in a fully CNN for deblurring spatially-variant blurred images. The utilization of this information showed improved results against the blind model in all tested cases, which indicates the effectiveness of leveraging such information. Moreover, to train our network, we proposed a novel technique to synthesize large amounts of spatially-variant blurred images using sharp RGB-D images and sampled 6D camera motions. Our synthetic dataset showed better results against single and stereo DAVANet [49] when deblurring real blurry images. Thus, our synthetic dataset is highly effective for enabling the model to generalize better to real-world blur.

Through our experiments, we trained the single DAVANet model with our synthetic data which revealed better prediction abilities against our blind model. The improvement is attributed to the utilization of residual blocks and the concatenation of convolution layer outputs with different dilation rates. Understanding the reasons behind the utilization of such components and the advantage they provide is essential for designing proper models to address the image deblurring problem.

In addition to our effective synthetic dataset, a promising multi-camera setup was proposed to collect real blurry and sharp image pairs and the corresponding scene depth and IMU measurements. While collecting such data is extremely challenging, we demonstrated motivating results for continuation of development in that direction. Several challenges hindered the process of synchronization and alignment of the blurry and sharp image pairs, especially the temporal shift and the rolling shutter effect of the camera.

Finetuning our pretrained model with the collected real dataset resulted in blur-like artefacts in the output. This means that the slight misalignments between the blurry image and the sharp reference image should be further handled to make the real dataset effective for training the deblurring models. Disregarding the misalignments, the finetuning process itself should be approached carefully.

Wide space exists for improvement and development in the ideas and techniques of image deblurring. Additionally, as the advances and developments in neural networks continue, more robust and effective learning models can be built and utilized for approaching the challenging problem of image deblurring.

9. REFERENCES

[1] Wang R. & Tao D. (2014) Recent progress in image deblurring. arXiv preprint arXiv:1409.6838 .

[2] of Dayton U., atmospheric turbulence. URL: `https://udayton.edu/engineering/research/research-labs/signal-and-image-processing-lab/_resources/img/atmospheric-turbulence-mitigation.jpg`.

[3] Whyte O., Sivic J., Zisserman A. & Ponce J. (2012) Non-uniform deblurring for shaken images. International journal of computer vision 98, pp. 168–186.

[4] Tai Y.W., Tan P. & Brown M.S. (2010) Richardson-lucy deblurring for scenes under a projective motion path. IEEE Transactions on Pattern Analysis and Machine Intelligence 33, pp. 1603–1618.

[5] Hu Z., Xu L. & Yang M.H. (2014) Joint depth estimation and camera shake removal from single blurry image. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2893–2900.

[6] Fergus R., Singh B., Hertzmann A., Roweis S.T. & Freeman W.T. (2006) Removing camera shake from a single photograph. In: ACM SIGGRAPH 2006 Papers, pp. 787–794.

[7] Levin A. (2007) Blind motion deblurring using image statistics. In: Advances in Neural Information Processing Systems, pp. 841–848.

[8] Shan Q., Jia J. & Agarwala A. (2008) High-quality motion deblurring from a single image. Acm transactions on graphics (tog) 27, pp. 1–10.

[9] Cho S. & Lee S. (2009) Fast motion deblurring. In: ACM SIGGRAPH Asia 2009 papers, pp. 1–8.

[10] Xu L., Zheng S. & Jia J. (2013) Unnatural l0 sparse representation for natural image deblurring. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1107–1114.

[11] Pan J., Sun D., Pfister H. & Yang M.H. (2016) Blind image deblurring using dark channel prior. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1628–1636.

[12] Lai W.S., Ding J.J., Lin Y.Y. & Chuang Y.Y. (2015) Blur kernel estimation using normalized color-line prior. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 64–72.

[13] Xu L. & Jia J. (2012) Depth-aware motion deblurring. In: 2012 IEEE International Conference on Computational Photography (ICCP), IEEE, pp. 1–8.

[14] Pan L., Dai Y. & Liu M. (2019) Single image deblurring and camera motion estimation with depth map. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, pp. 2116–2125.

[15] Joshi N., Kang S.B., Zitnick C.L. & Szeliski R. (2010) Image deblurring using inertial measurement sensors. ACM Transactions on Graphics (TOG) 29, pp. 1–9.

[16] Hu Z., Yuan L., Lin S. & Yang M.H. (2016) Image deblurring using smartphone inertial sensors. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1855–1864.

[17] Sindelar O. & Sroubek F. (2013) Image deblurring in smartphone devices using built-in inertial measurement sensors. Journal of Electronic Imaging 22, p. 011003.

[18] Mustaniemi J., Kannala J., Särkkä S., Matas J. & Heikkilä J. (2018) Fast motion deblurring for feature detection and matching using inertial measurements. In: 2018 24th International Conference on Pattern Recognition (ICPR), IEEE, pp. 3068–3073.

[19] Hee Park S. & Levoy M. (2014) Gyro-based multi-image deconvolution for removing handshake blur. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3366–3373.

[20] Tai Y.W., Du H., Brown M.S. & Lin S. (2008) Image/video deblurring using a hybrid camera. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp. 1–8.

[21] Li F., Yu J. & Chai J. (2008) A hybrid camera for motion deblurring and depth map super-resolution. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp. 1–8.

[22] Yuan L., Sun J., Quan L. & Shum H.Y. (2007) Image deblurring with blurred/noisy image pairs. In: ACM SIGGRAPH 2007 papers, pp. 1–es.

[23] Park H. & Mu Lee K. (2017) Joint estimation of camera pose, depth, deblurring, and super-resolution from a blurred image sequence. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 4613–4621.

[24] Lee S. & Cho S. (2013) Recent advances in image deblurring. In: SIGGRAPH Asia 2013 Courses, pp. 1–108.

[25] Richardson W.H. (1972) Bayesian-based iterative method of image restoration. JoSA 62, pp. 55–59.

[26] Levin A., Weiss Y., Durand F. & Freeman W.T. (2009) Understanding and evaluating blind deconvolution algorithms. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, pp. 1964–1971.

[27] Lai W.S., Huang J.B., Hu Z., Ahuja N. & Yang M.H. (2016) A comparative study for single image blind deblurring. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1701–1709.

[28] Gupta A., Joshi N., Zitnick C.L., Cohen M. & Curless B. (2010) Single image deblurring using motion density functions. In: European Conference on Computer Vision, Springer, pp. 171–184.

[29] Ruskai M., Beylkin G., Coifman R. & Daubechies I. (1994) Lb lucy," an iterative technique for the rectification of observed distributions", astronomical journal, vol. 79, 1974, 745-754. s. mallat," a theory for multiresolution signal decomposition: the wavelet representation," ieee transac-tions on pattern analysis and machine intelligence, vol. 11, 1989, pp. 674-693. y. meyer, proc. ondelettes el paquets d'ondes, inria, rocquencourt, 1991. Image Reconstruction and Restoration 2302, p. 142.

[30] Hirsch M., Schuler C.J., Harmeling S. & Schölkopf B. (2011) Fast removal of non-uniform camera shake. In: 2011 International Conference on Computer Vision, IEEE, pp. 463–470.

[31] Stanford U., Neural networks. URL: `https://cs231n.github.io/neural-networks-1/`.

[32] Goodfellow I., Bengio Y. & Courville A. (2016) Deep learning. MIT press.

[33] Rumelhart D.E., Hinton G.E. & Williams R.J. (1986) Learning representations by back-propagating errors. nature 323, pp. 533–536.

[34] Kingma D.P. & Ba J. (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 .

[35] LeCun Y., Kavukcuoglu K. & Farabet C. (2010) Convolutional networks and applications in vision. In: Proceedings of 2010 IEEE international symposium on circuits and systems, IEEE, pp. 253–256.

[36] LeCun Y., Bottou L., Bengio Y. & Haffner P. (1998) Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, pp. 2278–2324.

[37] Ronneberger O., Fischer P. & Brox T. (2015) U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention, Springer, pp. 234–241.

[38] Nair V. & Hinton G.E. (2010) Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp. 807–814.

[39] Dumoulin V. & Visin F. (2016), A guide to convolution arithmetic for deep learning.

[40] Yu F. & Koltun V. (2015) Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 .

[41] Lin M., Chen Q. & Yan S. (2013), Network in network.

[42] Dai J., Qi H., Xiong Y., Li Y., Zhang G., Hu H. & Wei Y. (2017) Deformable convolutional networks. In: Proceedings of the IEEE international conference on computer vision, pp. 764–773.

[43] He K., Zhang X., Ren S. & Sun J. (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778.

[44] Isola P., Zhu J.Y., Zhou T. & Efros A.A. (2017) Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1125–1134.

[45] Kupyn O., Budzan V., Mykhailych M., Mishkin D. & Matas J. (2018) Deblurgan: Blind motion deblurring using conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 8183–8192.

[46] Tao X., Gao H., Shen X., Wang J. & Jia J. (2018) Scale-recurrent network for deep image deblurring. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8174–8182.

[47] Nah S., Hyun Kim T. & Mu Lee K. (2017) Deep multi-scale convolutional neural network for dynamic scene deblurring. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3883–3891.

[48] Zhang J., Pan J., Ren J., Song Y., Bao L., Lau R.W. & Yang M.H. (2018) Dynamic scene deblurring using spatially variant recurrent neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2521–2529.

[49] Zhou S., Zhang J., Zuo W., Xie H., Pan J. & Ren J.S. (2019) Davanet: Stereo deblurring with view aggregation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 10996–11005.

[50] Mustaniemi J., Kannala J., Särkkä S., Matas J. & Heikkila J. (2019) Gyroscope-aided motion deblurring with deep networks. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, pp. 1914–1922.

[51] Sun J., Cao W., Xu Z. & Ponce J. (2015) Learning a convolutional neural network for non-uniform motion blur removal. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 769–777.

[52] Gong D., Yang J., Liu L., Zhang Y., Reid I., Shen C., Van Den Hengel A. & Shi Q. (2017) From motion blur to motion flow: a deep learning solution for removing heterogeneous motion blur. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2319–2328.

[53] Xu X., Pan J., Zhang Y.J. & Yang M.H. (2017) Motion blur kernel estimation via deep learning. IEEE Transactions on Image Processing 27, pp. 194–205.

[54] Chakrabarti A. (2016) A neural approach to blind motion deblurring. In: European conference on computer vision, Springer, pp. 221–235.

[55] Schuler C.J., Hirsch M., Harmeling S. & Schölkopf B. (2015) Learning to deblur. IEEE transactions on pattern analysis and machine intelligence 38, pp. 1439–1451.

[56] Wang R. & Tao D. (2018) Training very deep cnns for general non-blind deconvolution. IEEE Transactions on Image Processing 27, pp. 2897–2910.

[57] Son H. & Lee S. (2017) Fast non-blind deconvolution via regularized residual networks with long/short skip-connections. In: 2017 IEEE International Conference on Computational Photography (ICCP), IEEE, pp. 1–10.

[58] Zhang J., Pan J., Lai W.S., Lau R.W. & Yang M.H. (2017) Learning fully convolutional networks for iterative non-blind deconvolution. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3817–3825.

[59] Noroozi M., Chandramouli P. & Favaro P. (2017) Motion deblurring in the wild. In: German conference on pattern recognition, Springer, pp. 65–77.

[60] Gao H., Tao X., Shen X. & Jia J. (2019) Dynamic scene deblurring with parameter selective sharing and nested skip connections. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3848–3856.

[61] Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A. & Bengio Y. (2014) Generative adversarial nets. In: Advances in neural information processing systems, pp. 2672–2680.

[62] Ulyanov D., Vedaldi A. & Lempitsky V. (2016) Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 .

[63] Mustaniemi J., Kannala J., Matas J., Särkkä S. & Heikkilä J. (2018) Lsd _2-joint denoising and deblurring of short and long exposure images with convolutional neural networks. arXiv preprint arXiv:1811.09485 .

[64] Aittala M. & Durand F. (2018) Burst image deblurring using permutation invariant convolutional neural networks. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 731–747.

[65] Su S., Delbracio M., Wang J., Sapiro G., Heidrich W. & Wang O. (2017) Deep video deblurring for hand-held cameras. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1279–1288.

[66] Kim T.H., Sajjadi M.S., Hirsch M. & Schölkopf B. (2018) Spatio-temporal transformer network for video restoration. In: European Conference on Computer Vision, Springer, pp. 111–127.

[67] Hyun Kim T., Mu Lee K., Scholkopf B. & Hirsch M. (2017) Online video deblurring via dynamic temporal blending network. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 4038–4047.

[68] Chen H., Gu J., Gallo O., Liu M.Y., Veeraraghavan A. & Kautz J. (2018) Reblur2deblur: Deblurring videos via self-supervised learning. In: 2018 IEEE International Conference on Computational Photography (ICCP), IEEE, pp. 1–9.

[69] Zhou S., Zhang J., Pan J., Xie H., Zuo W. & Ren J. (2019) Spatio-temporal filter adaptive network for video deblurring. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2482–2491.

[70] Johnson J., Alahi A. & Fei-Fei L. (2016) Perceptual losses for real-time style transfer and super-resolution. In: European conference on computer vision, Springer, pp. 694–711.

[71] Simonyan K. & Zisserman A. (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 .

[72] Behnel S., Bradshaw R., Citro C., Dalcin L., Seljebotn D.S. & Smith K. (2011) Cython: The best of both worlds. Computing in Science & Engineering 13, pp. 31–39.

[73] Bradski G. (2000) The OpenCV Library. Dr. Dobb's Journal of Software Tools .

[74] McCormac J., Handa A., Leutenegger S. & Davison A.J. (2016) Scenenet rgb-d: 5m photorealistic images of synthetic indoor trajectories with ground truth. arXiv preprint arXiv:1612.05079 .

[75] StereoLabs, Zed camera. URL: `https://www.stereolabs.com/zed/`.

[76] Lowe D.G. (2004) Distinctive image features from scale-invariant keypoints. International journal of computer vision 60, pp. 91–110.

[77] Fischler M.A. & Bolles R.C. (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24, pp. 381–395.

[78] Wang Z., Bovik A.C., Sheikh H.R. & Simoncelli E.P. (2004) Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing 13, pp. 600–612.

[79] Sheikh H.R. & Bovik A.C. (2006) Image information and visual quality. IEEE Transactions on image processing 15, pp. 430–444.