ACCELERATION OF NON-LINEAR IMAGE FILTERS, AND MULTI-FRAME IMAGE DENOISING

Dissertation

Submitted to

The School of Engineering of the

UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for

The Degree of

Doctor of Philosophy in Engineering

By

Christina Maria Karam

Dayton, Ohio

December, 2019

ACCELERATION OF NON-LINEAR IMAGE FILTERS, AND MULTI-FRAME IMAGE DENOISING

Name: Karam, Christina Maria

APPROVED BY:

Keigo Hirakawa, Ph.D. Advisory Committee Chairman Associate Professor, Electrical and Computer Engineering	Eric J. Balster, Ph.D. Committee Member Associate Professor and Chair, Electrical and Computer Engineering
Raúl Ordoñez, Ph.D. Committee Member Professor, Electrical and Computer Engineering	Ju Shen, Ph.D. Committee Member Assistant Professor, Computer Science
Robert J. Wilkens, Ph.D., P.E. Associate Dean for Research and Innovation Professor School of Engineering	Eddy M. Rojas, Ph.D., M.A., P.E. Dean, School of Engineering

ABSTRACT

ACCELERATION OF NON-LINEAR IMAGE FILTERS, AND MULTI-FRAME IMAGE DENOISING

Name: Karam, Christina Maria
University of Dayton

Advisor: Dr. Keigo Hirakawa

This dissertation is comprised of four novel contributions. First, we propose new implementations of Monte-Carlo-based bilateral filter and non-local means whose per-pixel complexity is approximately invariant to the color dimension, window size, and block size. We reduce complexity by combining the random filtering of multiple color channels that approximate the non-linear behavior of the bilateral filter into a single convolution operation. We extend this work to a non-linear filter called Non-Local Means. In the second part, we propose "convolutional distance transform"—efficient implementations of distance transform. Specifically, we leverage approximate minimum functions to rewrite the distance transform in terms of convolution operators, reducing the complexity to $N \log N$. Third, we propose a novel method for multi-frame image denoising in mobile phones. We developed a method to register noisy image frames by estimating the camera motion using both image and inertial measurements. Lastly, we develop a new framework for multi-frame image denoising using noisy image statistics of one frame to design an optimal denoising filter for the second frame. The algorithm is provably optimal in minimum mean squared error estimation sense as well as in wavelet structural similarity metric sense.

For Salim, Betty, Karam and Paul

ACKNOWLEDGMENTS

There is a long list of people I would like to thank for making all this work possible, and I will **try** to keep it as concise as possible.

I would like to thank the Graduate School at UD and the Samsung research team for funding my projects, my committee members Dr. Raúl Ordoñez, Dr. Eric Balster, Dr. Ju Shen for their guidance throughout these past few years, and Kenjiro Sugimoto for his collaboration in parts of the work.

To Dr. Keigo Hirakawa: your guidance, your support and your encouragement since I've joined your lab are unmatched. Without you, this dissertation would not have been completed. You were understanding during difficult times, and ever-present in time of need. You've given me opportunities I would never have dreamed of which have allowed me to meet the most incredible people.

This brings me to my labmates: you're the funnest group of people anyone will ever get the chance to work with. You've turned long work hours into a fun and entertaining time, you've turned the lab into a "jungle" (as some would say), and you've graced me with quotes I will never forget. Some of you still owe me slingshot chickens, though.

To my friends, both near and far: you believed in me when I didn't believe in myself, you've encouraged me and you've been supportive from start to finish. Despite helping me procrastinate, the emotional support I've received from all of you has gotten me to where I am. Thanks for the food in time of need, the late nights hanging out, the heart to hearts, and the distractions that were very much needed.

To my family spread out around the world: you are the reason I am who I am today. You are the best part of me and I've learned so much from each and every one of you.

Last but definitely not least, to mom and dad: you've taught me what hard work and perseverance is. The sacrifices you've made have allowed Karam, Paul and I to get to where we are, and I hope this makes you proud.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

1.1 Bilateral Filter and Non-Local Means

Bilateral filter (BF) is an image smoothing filtering technique introduced by [7–9]. Image features such as textures and edges are preserved by BF due to the adaptive weighting of the spatial and range kernels. The former assigns higher weights to the spatially nearby pixels, while the latter gives more importance to the pixels of similar appearance. Despite the proven usefulness of BF in many image processing and computer vision applications—as evidenced by prior examples in denoising [10], demosaicking [11], tone mapping [12], stereo matching [13], and segmentation [14]—the complexity of BF remains a limiting factor.

Non-local means (NLM) is a generalization of the bilateral filter that has shown advantages in denoising [15]. It replaces the notion of pixel-to-pixel similarity [15] in range kernel with a block-to-block similarity. While edges and textures are better preserved by the block extension of the range kernel, the computational complexity increases significantly. Although there are arguably superior denoising techniques available [16], NLM remains popular today owing to its intuitiveness, filtering quality, and suitability for working with color images.

We previously introduced stochastic bilateral filter (SBF) and stochastic non-local means (SNLM) [17], two implementations aimed at speeding up bilateral filter and non-local means, respectively. Specifically, SBF/SNLM replaced the range kernel computation by the efficient randomized convolutional processes that agree with BF/NLM *on average* by the way of Monte-Carlo process. As an implementation whose complexity and Monte-Carlo convergence rate are largely invariant to the color dimension of the edge image, the window size,

and the block sizes, SBF and SNLM are orders of magnitude faster than the other existing "fast" implementations [2–4, 6, 18–22].

Yet, the complexities of SBF and SNLM in [17] grow *linearly* with respect to the color dimension of the filter image. In this paper, we propose a technique to accelerate SBF and SNLM further by combining the random filtering of multiple color channels into a single random filtering process. The complexities of the resultant fast stochastic bilateral filter (FSBF) and fast stochastic non-local means (FSNLM) implementations grow very slowly with respect to the color dimension of the filtering image than the existing methods. That is, they allow the processing of high-dimensional images (such as hyperspectral images) with only a modest complexity.

Simplified versions of Lemmas 1 and 2, Theorem 1, and Corollary 1 have appeared in our preliminary work in [23]. In this dissertation, more general versions are re-derived in order to accommodate the NLM weights in (2.5) below (subsequently used to derive the FSNLM results in Lemma 3 and Theorem 2). We also point out one limitation to this work, which is that FSBF/FSNLM cannot use an edge image separate from the filtering image (i.e. no cross-bilateral filtering). It is also worth emphasizing that the work presented in this dissertation is not meant to improve the filtering qualities of BF or NLM. Rather, we are only addressing the complexity issues of BF and NLM, particularly for high-dimensional data.

1.2 Distance Transform

A distance transform is a grayscale image in which the intensity level shows the distance of each pixel to the nearest "edge" pixel. It can also be interpreted as a surface whose height is proportional to the distances (i.e. iso-distance curves). The distance transform is

commonly used in computer vision and image processing applications for the separation of overlapping objects [24,25], robot navigation [25–27], skeletonization [28–31], shape analysis [32–35], and segmentation [36]. Despite the usefulness of the distance transform, its overall complexity is $\mathcal{O}(N^2)$ (or $\mathcal{O}(N)$ "per pixel" complexity), where N is the image size, making it slow for a large image.

In this dissertation, we propose three different approximations to the distance transform (Theorems 3–5 below), aimed at overcoming the complexity issue. They are based on convolutions, reducing the complexity to $\mathcal{O}(N \log N)$ (or $\mathcal{O}(\log N)$ per pixel) if implemented using a fast Fourier transform (FFT). In special cases, we may further leverage separable convolutions (Corollary 4 and 5) and exploit constant-time Gaussian convolutions approximations ($\mathcal{O}(N)$). Unlike the prior art, our method generalizes to *any* translation-invariant metric distance transforms (i.e. not limited to p-norm or Euclidean distance) with $\mathcal{O}(N \log N)$ complexity. Our method is straightforward to implement, and generalizes to $L > 2$ dimensions (e.g. voxels or video signal) with minimal effort. We develop our technique in the discrete image signal, but the idea is equally applicable to the continuous image case.

1.3 Multi-Frame Image Denoising Using IMU

Nowadays, smartphones are equipped with inertial measurement units (IMUs) that allow us to track the relative movement of the device. These IMUs are inertial sensors, such as gyroscopes and accelerometers that have proven useful for applications in navigation, augmented and virtual reality, smart health, and more. Pairing these IMUs with a camera, the inertial measurements can be used for the deblurring of images by reconstructing the camera motion causing the blur, panoramic imaging, video stabilization, and correction for rolling shutters. Another problem the use of IMUs can help overcome is multi-frame

image denoising, where the inertial measurements will be used for image registration. These frames are combined to yield an estimated noise-free image.

Mobile phone cameras are very susceptible to noise due to the small size of the sensors that reduce the light efficiency. The sensors are even smaller in high resolution images making the problem even worse. Increasing the exposure time is one way to overcome this problem in low light imaging. However, this increases the chances of image blur caused my camera shake during the duration of the exposure. A short exposure, on the other hand, would yield a very noisy image and even though image denoising can be somewhat helpful, the high spatial frequency details will be lost, giving the image a smoother appearance. This limits how short the exposure time can be.

To overcome the blur caused by long exposure and the noise caused by a shorter exposure, the alternative is multi-frame image denoising. This is done by taking N short exposure images in a burst. These N images are then registered[1] and then, once registered, they are combined to yield one denoised image. The advantages of this approach are twofold: first, each image has a short exposure time minimizing the risk of blur, and second, the noise in each image is independent, which allows the reconstruction of the high spatial frequency details that is not possible in a single frame image denoising approach. However, there are also challenges with this approach. First, objects within the scenes are not necessarily stationary, and the resulting combined image might have motion artifacts. Second, image registration of very noisy images is difficult because the scene content is less clear, and finally, the registration errors lead to a loss of image resolution because combining displaced pixels attenuates the high frequency components.

[1]Image registration is a task of re-positioning multiple frames to the same coordinates system so that a particular pixel position in the registered images represents the same point on the scene.

Incorporating inertial measurements in multi-frame image denoising helps in overcoming the issues of image registration. The gyroscope measurements allow a reconstruction of the rough trajectory of the camera, which can be used to reposition each frame. In the case of multi-frame imaging, estimating the camera motion from the IMU meausurements is more reliable than estimating that motion from the image content, because the IMUs are not influenced by the lighting conditions. However, IMUs are not perfect and suffer from noise and drift. Another challenge is that cameras and IMUs are not perfectly synchronized [37] and are not aligned, particularly in smartphones.

However, due to hardware limitations, image noise is almost inevitable in low light imaging with short exposure time. Image denoising is one way of removing the undesired signal. Many denoising algorithms require a statistical model of the signal corruption which is not always easy to determine. Some methods, such as block matching and 3D filtering (BM3D) [38] and non-local means (NLM) [39] compare blocks of pixels within an image to each other, and leverage information from similar blocks to filter the image. Noise2Noise is a method that allows us to obtain clean images while only having access to noisy images [40, 41], irrespective of the noise model. It has proven useful by training neural networks to learn the statistical model of noise corruption indirectly from the data, as well as in MRI reconstructions from undersampled data [42]. It's also been useful in applications using pairs of noisy images, such as astrophotography, where typically, a long exposure is preferred. Albeit proven useful in practice, the claim presented in [42] has never been theoretically supported.

In this dissertation, we propose a novel solution for combining IMU and camera sensors for image denoising. The technique relies on the idea that camera motion is encoded in both the displacement of frames, as well as that of the inertial measurements. Leveraging

information from both, we infer the camera trajectory. The anticipated result is that the image registration of our proposed approach would yield better result than the image registration from image or IMU alone, because the noisy frame data would be compensated by the IMU, and vice versa. We also propose six Noise2Noise theorems that concretely prove the intuitive claim proposed in [42]: finding the variance, the mean squared error, the structural similarity (SSIM) and its wavelet variation (WSSIM), as well as the minimum mean squared error (MMSE) and the denoising wavelet structural similarity that are used to clean the image. These two different Noise2Noise denoising approaches use a minimum of two or three corrupted images –based on the algorithm– to obtain one clean output.

CHAPTER II

BACKGROUND: BILATERAL FILTER AND NON-LOCAL MEANS

2.1 Background and Related Works

2.1.1 Bilateral Filter and Non-Local Means

Let $\boldsymbol{f} : \mathbb{Z}^2 \to \mathbb{R}^C$ be the input filter image, where $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^C$ is a color vector at the pixel position $\boldsymbol{x} \in \mathbb{Z}^2$. The bilateral filter yields the output image $\boldsymbol{g} : \mathbb{Z}^2 \to \mathbb{R}^C$ defined by the relation:

$$\boldsymbol{g}\{\boldsymbol{f}\}(\boldsymbol{x}) := \frac{\sum\limits_{\boldsymbol{y}\in\mathbb{Z}^2} w(\boldsymbol{x}-\boldsymbol{y}) \cdot \phi(\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y})) \cdot \boldsymbol{f}(\boldsymbol{y})}{\sum\limits_{\boldsymbol{y}\in\mathbb{Z}^2} w(\boldsymbol{x}-\boldsymbol{y}) \cdot \phi(\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y}))}. \tag{2.1}$$

where the spatial kernel $w : \mathbb{Z}^2 \to \mathbb{R}$ and range kernel $\phi : \mathbb{R}^C \to \mathbb{R}$ are defined as

$$\begin{aligned} w(\boldsymbol{x}-\boldsymbol{y}) &:= \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\sigma^2}\right) \\ \phi(\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y})) &:= \exp\left(-\frac{(\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y}))^T \Theta^{-1} (\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y}))}{2}\right). \end{aligned} \tag{2.2}$$

Here, $\|\cdot\|$ denotes the ℓ^2 norm in $\mathbb{Z}^2$, and $\sigma^2 \in \mathbb{R}$ and $\Theta \in \mathbb{R}^{C\times C}$ are smoothing parameters. Intuitively, bilateral filter preserves edges by ensuring w and ϕ are small when $\|\boldsymbol{x}-\boldsymbol{y}\|$ and/or $(\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y}))^T \Theta^{-1} (\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y}))$ are large. Although Θ in most cases is a constant diagonal matrix of the form

$$\Theta = \begin{bmatrix} \theta^2 & & \\ & \ddots & \\ & & \theta^2 \end{bmatrix}, \tag{2.3}$$

which gives equal importance to each color component, we use the general form of Θ later in this dissertation. Cross-color correlation can be considered by introducing off-diagonal elements in Θ.

Recall that the bilateral pixel range kernel determines the weight of the linear combination in (2.1) based on the pixel-to-pixel similarity of $\boldsymbol{x}$ and $\boldsymbol{y}$ [7–9]. The non-local means

generalizes this by computing the linear weight based on the similarity of pixel blocks centered at $\boldsymbol{x}$ and $\boldsymbol{y}$. Define $\boldsymbol{h} : \mathbb{Z}^2 \to \mathbb{R}^2$ be the result of filtering the input image $\boldsymbol{f}$ using the relation:

$$\boldsymbol{h}\{\boldsymbol{f}\}(\boldsymbol{x}) := \frac{\sum_{\boldsymbol{y}\in\mathbb{Z}^2} w(\boldsymbol{x}-\boldsymbol{y}) \cdot \psi\{\boldsymbol{f}\}(\boldsymbol{x},\boldsymbol{y}) \cdot \boldsymbol{f}(\boldsymbol{y})}{\sum_{\boldsymbol{y}\in\mathbb{Z}^2} w(\boldsymbol{x}-\boldsymbol{y}) \cdot \psi\{\boldsymbol{f}\}(\boldsymbol{x},\boldsymbol{y})} \tag{2.4}$$

where NLM range kernel $\psi : \mathbb{Z}^2 \times \mathbb{Z}^2 \to \mathbb{R}$ is defined as

$$\psi\{\boldsymbol{f}\}(\boldsymbol{x},\boldsymbol{y}) := \exp\left(-\sum_{\boldsymbol{b}\in\Gamma} \frac{(\boldsymbol{f}(\boldsymbol{x}-\boldsymbol{b}) - \boldsymbol{f}(\boldsymbol{y}-\boldsymbol{b}))^T \Theta_{\boldsymbol{b}}^{-1} (\boldsymbol{f}(\boldsymbol{x}-\boldsymbol{b}) - \boldsymbol{f}(\boldsymbol{y}-\boldsymbol{b}))}{2}\right) \tag{2.5}$$

and $\Gamma \subset \mathbb{Z}^2$ indicates the pixel indexes in the $B \times B$ block. In other words, pixel similarity of $\boldsymbol{f}(\boldsymbol{x}-\boldsymbol{b})$ and $\boldsymbol{f}(\boldsymbol{x}-\boldsymbol{b})$ for all $\boldsymbol{b} \in \Gamma$ are considered jointly to determine the weight $\psi\{\boldsymbol{f}\}(\boldsymbol{x},\boldsymbol{y})$. The usual choice for the smoothing parameter $\Theta_{\boldsymbol{b}} \in \mathbb{R}^{C\times C}$ is a constant diagonal matrix:

$$\Theta_{\boldsymbol{b}} = \begin{bmatrix} \theta_{\boldsymbol{b}}^2 & & \\ & \ddots & \\ & & \theta_{\boldsymbol{b}}^2 \end{bmatrix}, \tag{2.6}$$

but within-block weights $\theta_{\boldsymbol{b}}^2$ can be varied optionally to increase the influence of the pixels at the center of the block.

The complexities of the bilateral filter and non-local means are reported in Tables 3.1 and 3.2, respectively. The bottleneck is highlighted in red. Thanks to the fact that $w(\boldsymbol{x}-\boldsymbol{y})$ decays quickly relative to increasing $\|\boldsymbol{x}-\boldsymbol{y}\|^2$, it is common to limit the summation in (2.1) and (2.4) to a spatial neighborhood of the window size $W \times W$. Hence the overall per-pixel complexity is $\mathcal{O}(W^2C)$ for BF and $\mathcal{O}(W^2B^2C)$ for NLM. Previous efforts to accelerate BF have largely focused on developing implementations that whose complexities are invariant to the window size W^2 and block size B^2. The dependence of BF complexity on W^2 and B^2 is unattractive because the resolutions of modern imaging devices are increasing rapidly—a larger window/block neighborhood around the pixel $\boldsymbol{x}$ is needed to represent

the same underlying image feature near $\boldsymbol{x}$. Indeed, the complexity of "fast" bilateral filter implementations of [2–4, 6, 18, 19] shown in Table 3.1 are constant with respect to W. However, the implementation complexity of $\mathcal{O}(CK^C)$ in [2,18] and $\mathcal{O}(K^C)$ in [19] for some constant K is unacceptably large unless $C = 1$ (i.e. $f : \mathbb{Z}^2 \to \mathbb{R}$ is a grayscale image). More recent treatments of BF (including SBF) further reduced the complexity (as determined by the number of convolution operators) to $\mathcal{O}(CK)$ [3,4]. Similarly, earlier efforts to accelerate NLM resulted in only a slight improvement at $\mathcal{O}(W^2B^2CP)$ where $0 < P < 1$ [20, 21]. Recent efforts reduced the complexity to $\mathcal{O}(W^2C)$ [22] and $\mathcal{O}(CK)$ in [17]. By contrast, the new BF/NLM implementations developed in this dissertation achieves the per-pixel complexity of $\mathcal{O}(K)$, i.e. independent of W, B, and C.

2.1.2 Prior Work on Accelerated Filters

To process high-dimensional images in a reasonable amount of time, improving the speed is key. Stochastic Bilateral Filter (SBF) and Stochastic Non-Local Means (SNLM) overcome the tight coupling between the complexity of bilateral filtering and the color dimension C using randomized filters [17]. Below, the filters $\widetilde{g}\{\boldsymbol{f}\}(\boldsymbol{x})$ and $\widetilde{h}\{\boldsymbol{f}\}(\boldsymbol{x})$ agree with the bilateral filter $g\{\boldsymbol{f}\}(\boldsymbol{x})$ and non-local means $h\{\boldsymbol{f}\}(\boldsymbol{x})$ results, respectively.

Proposition 1 (SBF)**.** *Let $\boldsymbol{\zeta} \sim \mathcal{N}(0, \Theta^{-1})$, $\boldsymbol{\zeta} \in \mathbb{R}^C$ be a normal random vector. Define*

$$\widetilde{\boldsymbol{g}}\{\boldsymbol{f}\}(\boldsymbol{x}) := \frac{\mathbb{E}\left[\begin{bmatrix}\cos(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{x}))\\ \sin(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{x}))\end{bmatrix}^T\left(w(\boldsymbol{x}) \star \left\{\boldsymbol{f}(\boldsymbol{x})\begin{bmatrix}\cos(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{y}))\\ \sin(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{y}))\end{bmatrix}\right\}\right)\right]}{\mathbb{E}\left[\begin{bmatrix}\cos(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{x}))\\ \sin(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{x}))\end{bmatrix}^T\left(w(\boldsymbol{x}) \star \begin{bmatrix}\cos(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{y}))\\ \sin(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{y}))\end{bmatrix}\right)\right]}. \tag{2.7}$$

Then $\widetilde{g}\{\boldsymbol{f}\}(\boldsymbol{x})$ is equivalent to $\boldsymbol{g}\{\boldsymbol{f}\}(\boldsymbol{x})$ in (2.1)*.*

Proposition 2 (SNLM)**.** *Let* $\boldsymbol{\zeta}(\boldsymbol{b}) \sim \mathcal{N}(0, \Theta_{\boldsymbol{b}}^{-1})$, $\boldsymbol{\zeta}(\boldsymbol{b}) \in \mathbb{R}^C$, *be independent random vectors defined over* $\boldsymbol{b} \in \boldsymbol{\Gamma}$. *Define convolution-sum operator* $\circledast$ *as*

$$\{\boldsymbol{\zeta} \circledast \boldsymbol{f}\}(\boldsymbol{x}) := \sum_{\boldsymbol{b} \in \Gamma} \boldsymbol{\zeta}^T(\boldsymbol{b}) \boldsymbol{f}(\boldsymbol{x} - \boldsymbol{b}). \tag{2.8}$$

The SNLM $\widetilde{\boldsymbol{h}}\{\boldsymbol{f}\}(\boldsymbol{x})$ *defined as*

$$\widetilde{\boldsymbol{h}}\{\boldsymbol{f}\}(\boldsymbol{x}) = \frac{\mathbb{E}\left[\begin{bmatrix}\cos(\boldsymbol{\zeta} \circledast \boldsymbol{f}) \\ \sin(\boldsymbol{\zeta} \circledast \boldsymbol{f})\end{bmatrix}^T \left(w(\boldsymbol{x}) \star \left\{ \boldsymbol{f}(\boldsymbol{x}) \begin{bmatrix}\cos(\boldsymbol{\zeta} \circledast \boldsymbol{f}) \\ \sin(\boldsymbol{\zeta} \circledast \boldsymbol{f})\end{bmatrix} \right\} \right) \right]}{\mathbb{E}\left[\begin{bmatrix}\cos(\boldsymbol{\zeta} \circledast \boldsymbol{f}) \\ \sin(\boldsymbol{\zeta} \circledast \boldsymbol{f})\end{bmatrix}^T \left(w(\boldsymbol{x}) \star \begin{bmatrix}\cos(\boldsymbol{\zeta} \circledast \boldsymbol{f}) \\ \sin(\boldsymbol{\zeta} \circledast \boldsymbol{f})\end{bmatrix} \right) \right]} \tag{2.9}$$

is also equivalent to NLM $\boldsymbol{h}\{\boldsymbol{f}\}(\boldsymbol{x})$ *in* (2.4).

Proof of Propositions 1 and 2 are provided in [17]. In practice, the range kernel is approximated by Monte-Carlo. A random vector $\boldsymbol{\zeta} \sim \mathcal{N}(0, \Theta^{-1})$ is generated L times, and then averaged to obtain an approximation of the expected value $\mathbb{E}[\cdot]$. It was proven in [17] and in Corollary 1 below that the convergence rate of the Monte-Carlo averaging is invariant to W, B, and C. Overall complexity of SBF/SNLM as determined by the number of convolutions is $O(CL)$, independent of W and B but scales linearly C and iteration number L. Using efficient implementations of Gaussian filters [43] the per-pixel complexity of convolution itself is $\mathcal{O}(1)$, invariant to W. As described in [17], the complexity of convolution-sum in (2.8) is invariant to B and C, and comparable to a conventional convolution using fast Fourier transform (FFT). See Tables 3.1 and 3.2.

CHAPTER III

PROPOSED METHOD: FAST STOCHASTIC BILATERAL FILTER AND NON-LOCAL MEANS

3.1 Proposed Stochastic Filters

3.1.1 Fast Stochastic Bilateral Filter

In this section, we aim to reduce the number of convolution operators by leveraging the fast compressive bilateral filtering in [19]. We begin by extending a key result in [19] to the color (i.e. $C > 1$) bilateral filter version, as follows.

Lemma 1. *Let* $\boldsymbol{q} = (q_1, \ldots, q_C)^T \in \mathbb{R}^C$. *Define the gradient of a range kernel* $\phi(\boldsymbol{q})$ *as:*

$$\nabla\phi(\boldsymbol{q}) = \begin{bmatrix} \frac{\partial}{\partial z_1}\phi(\boldsymbol{q}) \\ \vdots \\ \frac{\partial}{\partial z_C}\phi(\boldsymbol{q}) \end{bmatrix}. \tag{3.1}$$

Then $\boldsymbol{g}\{\boldsymbol{f}\}(\boldsymbol{x})$ *in* (2.1) *can be rewritten as:*

$$\boldsymbol{g}\{\boldsymbol{f}\}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}) + \Theta\frac{\sum\limits_{\boldsymbol{y}\in\mathbb{Z}^2} w(\boldsymbol{x}-\boldsymbol{y})\cdot\nabla\phi(\boldsymbol{f}(\boldsymbol{x})-\boldsymbol{f}(\boldsymbol{y}))}{\sum\limits_{\boldsymbol{y}\in\mathbb{Z}^2} w(\boldsymbol{x}-\boldsymbol{y})\cdot\phi(\boldsymbol{f}(\boldsymbol{x})-\boldsymbol{f}(\boldsymbol{y}))}. \tag{3.2}$$

Proof. Consider the difference image of the form:

$$\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{g}\{\boldsymbol{f}\}(\boldsymbol{x}) = \frac{\sum\limits_{\boldsymbol{y}\in\mathbb{Z}^2} w(\boldsymbol{x}-\boldsymbol{y})\cdot\phi(\boldsymbol{f}(\boldsymbol{x})-\boldsymbol{f}(\boldsymbol{y}))\cdot(\boldsymbol{f}(\boldsymbol{x})-\boldsymbol{f}(\boldsymbol{y}))}{\sum\limits_{\boldsymbol{y}\in\mathbb{Z}^2} w(\boldsymbol{x}-\boldsymbol{y})\cdot\phi(\boldsymbol{f}(\boldsymbol{x})-\boldsymbol{f}(\boldsymbol{y}))}. \tag{3.3}$$

Following recursion is a well-known property of Gaussian functions:

$$\nabla\phi(\boldsymbol{q}) = -\Theta^{-1}\boldsymbol{q}\exp\left(-\frac{\boldsymbol{q}^T\Theta^{-1}\boldsymbol{q}}{2}\right) = -\Theta^{-1}\boldsymbol{q}\phi(\boldsymbol{q}). \tag{3.4}$$

Substituting $\boldsymbol{q}\phi(\boldsymbol{q})$ by $-\Theta\nabla\phi(\boldsymbol{q})$ in (3.3) with $\boldsymbol{q} = \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y})$, we have

$$\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{g}\{\boldsymbol{f}\}(\boldsymbol{x}) = -\Theta\frac{\sum\limits_{\boldsymbol{y}\in\mathbb{Z}^2} w(\boldsymbol{x}-\boldsymbol{y})\cdot\nabla\phi(\boldsymbol{f}(\boldsymbol{x})-\boldsymbol{f}(\boldsymbol{y}))}{\sum\limits_{\boldsymbol{y}\in\mathbb{Z}^2} w(\boldsymbol{x}-\boldsymbol{y})\cdot\phi(\boldsymbol{f}(\boldsymbol{x})-\boldsymbol{f}(\boldsymbol{y}))}.$$

Reorganizing the above proves the Lemma. □

The prior work in [19] achieves BF complexity order of $\mathcal{O}(K^C)$ by combining Lemma 1 with the compressive bilateral filter in [18] (with complexity $\mathcal{O}(CK^C)$). In this work, we instead consider applying Lemma 1 to the random filtering of Propositions 1 and 2. Consider the following relation.

Lemma 2. *Let $\boldsymbol{\zeta} \sim \mathcal{N}(0, \Theta^{-1})$, $\boldsymbol{\zeta} \in \mathbb{R}^C$ be a normal random vector, and $\boldsymbol{q} \in \mathbb{R}^C$. Then*

$$\phi(\boldsymbol{q}) = \mathbb{E}\left[\cos\left(\boldsymbol{\zeta}^T\boldsymbol{q}\right)\right] \tag{3.5}$$

$$\nabla\phi(\boldsymbol{q}) = \mathbb{E}\left[-\boldsymbol{\zeta}\sin\left(\boldsymbol{\zeta}^T\boldsymbol{q}\right)\right]. \tag{3.6}$$

Proof. The proof of (3.5) is found in [17]. Recalling (3.1), we apply derivatives to the cosine functions in (3.5):

$$\nabla\phi(\boldsymbol{q}) = \mathbb{E}\begin{bmatrix}\frac{\partial}{\partial z_1}\cos(\boldsymbol{\zeta}^T\boldsymbol{q})\\ \vdots \\ \frac{\partial}{\partial z_C}\cos(\boldsymbol{\zeta}^T\boldsymbol{q})\end{bmatrix} = \mathbb{E}\begin{bmatrix}-\zeta_1\sin(\boldsymbol{\zeta}^T\boldsymbol{q})\\ \vdots \\ -\zeta_C\sin(\boldsymbol{\zeta}^T\boldsymbol{q})\end{bmatrix}. \tag{3.7}$$

Moving $\sin(\boldsymbol{\zeta}^T\boldsymbol{q})$ to outside of the matrix proves the Lemma. □

Combining Lemmas 1 and 2, we arrive at the proposed Fast Stochastic Bilateral Filter (FSBF) below.

Theorem 1 (FSBF)**.** *Let $\boldsymbol{\zeta} \sim \mathcal{N}(0, \Theta)$, $\boldsymbol{\zeta} \in \mathbb{R}^C$ be a normal random vector. Define*

$$\widehat{\boldsymbol{g}}\{\boldsymbol{f}\}(\boldsymbol{x}) := \boldsymbol{f}(\boldsymbol{x}) + \Theta\frac{\mathbb{E}\left[\boldsymbol{\zeta}\begin{bmatrix}-\sin(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{x}))\\ \cos(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{x}))\end{bmatrix}^T\left(w(\boldsymbol{x})\star\begin{bmatrix}\cos(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{x}))\\ \sin(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{x}))\end{bmatrix}\right)\right]}{\mathbb{E}\left[\begin{bmatrix}\cos(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{x}))\\ \sin(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{x}))\end{bmatrix}^T\left(w(\boldsymbol{x})\star\begin{bmatrix}\cos(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{x}))\\ \sin(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{x}))\end{bmatrix}\right)\right]}. \tag{3.8}$$

Then $\widehat{\boldsymbol{g}}\{\boldsymbol{f}\}(\boldsymbol{x})$ is equivalent to $\boldsymbol{g}\{\boldsymbol{f}\}(\boldsymbol{x})$ in (2.1).

Proof. Lemma 2 inspires separable representations of $\phi(\cdot)$ and $\nabla\phi(\cdot)$, respectively, as follows:

$$\begin{aligned}\phi(\boldsymbol{f}(\boldsymbol{x})-\boldsymbol{f}(\boldsymbol{y})) &= \mathbb{E}\left[\cos(\boldsymbol{\zeta}^T(\boldsymbol{f}(\boldsymbol{x})-\boldsymbol{f}(\boldsymbol{y})))\right] \qquad (3.9)\\ &= \mathbb{E}\left[\begin{bmatrix}\cos(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{x})) & \sin(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{x}))\end{bmatrix}\begin{bmatrix}\cos(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{y}))\\ \sin(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{y}))\end{bmatrix}\right].\\ \nabla\phi(\boldsymbol{f}(\boldsymbol{x})-\boldsymbol{f}(\boldsymbol{y})) &= \mathbb{E}[-\boldsymbol{\zeta}\sin(\boldsymbol{\zeta}^T(\boldsymbol{f}(\boldsymbol{x})-\boldsymbol{f}(\boldsymbol{y})))] \qquad (3.10)\\ &= \mathbb{E}\left[\boldsymbol{\zeta}\begin{bmatrix}-\sin(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{x})) & \cos(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{x}))\end{bmatrix}\begin{bmatrix}\cos(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{y}))\\ \sin(\boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{y}))\end{bmatrix}\right].\end{aligned}$$

Thus (3.8) follows from substituting (3.9) and (3.10) into Lemma 1. □

In practice, (3.8) is carried out by Monte-Carlo—as outlined in Algorithm 1, we approximate the expectation by drawing L random vectors $\boldsymbol{\zeta}\sim\mathcal{N}(0,\Theta)$, executing the computations inside the expectation operator, and averaging. FSBF in Theorem 1 is significantly less computationally intensive than SBF in Proposition 1 for two reasons. First, the convolutions in the numerator and the denominator of (3.8) are identical, meaning their computational burden can be shared. Second, the convolution in the numerator of (3.8) is C times less complex than the convolutions in the numerator of (2.7) because of the presence of $\boldsymbol{f}(\boldsymbol{x})\in\mathbb{R}^C$ in the latter case. As a result, the overall complexity (as determined by the number of convolutions) of the proposed FSBF reduces to $\mathcal{O}(L)$. The rate of Monte-Carlo convergence is described in Section 3.2.1 below.

3.1.2 Fast Stochastic Non-Local Means

We derive fast stochastic non-local means (FSNLM) as a generalization of fast stochastic bilateral filter. The relationship between the conventional non-local means and the conventional bilateral filter is made explicit in the Lemma 3. Below, let $\{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_{B^2}\}\in\Gamma$ refer to the indexes of the $B\times B$ block. The pixel index $m=(B^2+1)/2$ corresponds to the center position within the $B\times B$ block ("m" for *middle*). See Figure 3.1.

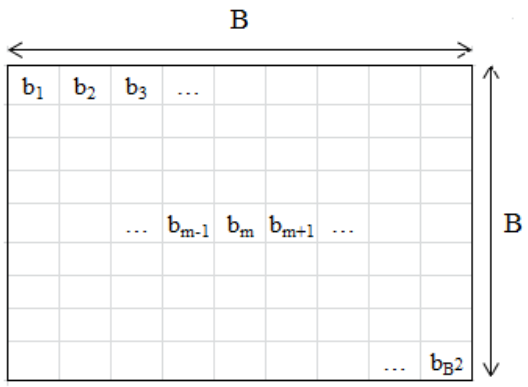


Figure 3.1: Pixel locations within a $B \times B$ block Γ.

Lemma 3. *Define* $\boldsymbol{e} : \mathbb{Z}^2 \to \mathbb{R}^{B^2C}$ *as:*

$$\boldsymbol{e}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{x} - \boldsymbol{b}_1) \\ \vdots \\ \boldsymbol{f}(\boldsymbol{x} - \boldsymbol{b}_{B^2}) \end{bmatrix}. \tag{3.11}$$

Set BF parameter $\Theta \in \mathbb{R}^{B^2C \times B^2C}$ *as a block diagonal matrix of the form:*

$$\Theta = \begin{bmatrix} \Theta_{\boldsymbol{b}_1} & & \\ & \ddots & \\ & & \Theta_{\boldsymbol{b}_{B^2}} \end{bmatrix}. \tag{3.12}$$

Suppose we partition the output of the bilateral filter $\boldsymbol{g}\{\boldsymbol{e}\}(\boldsymbol{x}) \in \mathbb{R}^{B^2C}$ *as*

$$\boldsymbol{g}\{\boldsymbol{e}\}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{g}_{\boldsymbol{b}_1}\{\boldsymbol{e}\}(\boldsymbol{x}) \\ \vdots \\ \boldsymbol{g}_{\boldsymbol{b}_{B^2}}\{\boldsymbol{e}\}(\boldsymbol{x}) \end{bmatrix}. \tag{3.13}$$

Then the NLM output $\boldsymbol{h}\{\boldsymbol{f}\}(\boldsymbol{x})$ *can be rewritten as:*

$$\boldsymbol{h}\{\boldsymbol{f}\}(\boldsymbol{x}) = \boldsymbol{g}_{\boldsymbol{b}_m}\{\boldsymbol{e}\}(\boldsymbol{x}). \tag{3.14}$$

Proof. Using the bilateral filter equation in (2.1), and the definition of $\boldsymbol{e}$ as (3.11), $\boldsymbol{g}_{\boldsymbol{b}}\{\boldsymbol{e}\}(\boldsymbol{x})$ takes the form:

$$\boldsymbol{g}_{\boldsymbol{b}}\{\boldsymbol{e}\}(\boldsymbol{x}) = \frac{\displaystyle\sum_{\boldsymbol{y} \in \mathbb{Z}^2} w(\boldsymbol{x} - \boldsymbol{y}) \cdot \phi(\boldsymbol{e}(\boldsymbol{x}) - \boldsymbol{e}(\boldsymbol{y})) \cdot \boldsymbol{f}(\boldsymbol{y} - \boldsymbol{b})}{\displaystyle\sum_{\boldsymbol{y} \in \mathbb{Z}^2} w(\boldsymbol{x} - \boldsymbol{y}) \cdot \phi(\boldsymbol{e}(\boldsymbol{x}) - \boldsymbol{e}(\boldsymbol{y}))}. \tag{3.15}$$

Plugging $\boldsymbol{e}(\cdot)$ in $\phi(\cdot)$, we get:

$$\begin{aligned}\phi(\boldsymbol{e}(\boldsymbol{x})-\boldsymbol{e}(\boldsymbol{y})) &= \exp\left(-\frac{(\boldsymbol{e}(\boldsymbol{x})-\boldsymbol{e}(\boldsymbol{y}))^T\Theta^{-1}(\boldsymbol{e}(\boldsymbol{x})-\boldsymbol{e}(\boldsymbol{y}))}{2}\right)\\ &= \exp\left(-\sum_{\boldsymbol{b}\in\Gamma}\frac{(\boldsymbol{e}(\boldsymbol{x}-\boldsymbol{b})-\boldsymbol{e}(\boldsymbol{y}-\boldsymbol{b}))^T\Theta_{\boldsymbol{b}}^{-1}(\boldsymbol{e}(\boldsymbol{x}-\boldsymbol{b})-\boldsymbol{e}(\boldsymbol{y}-\boldsymbol{b}))}{2}\right) \quad (3.16)\\ &= \psi\{\boldsymbol{e}\}(\boldsymbol{x},\boldsymbol{y}).\end{aligned}$$

Plugging (3.16) into (3.15) with $\boldsymbol{b}=\boldsymbol{b}_m$ proves the Lemma. □

Combining Lemma 3 and Theorem 1, we arrive at the proposed fast stochastic non-local means (FSNLM) implementation.

Theorem 2 (FSNLM)**.** *Let $\boldsymbol{\zeta}(\boldsymbol{b})\sim\mathcal{N}(0,\Theta_{\boldsymbol{b}}^{-1})$, $\boldsymbol{\zeta}(\boldsymbol{b})\in\mathbb{R}^C$, be independent random vectors defined over $\boldsymbol{b}\in\boldsymbol{\Gamma}$. Define $\widehat{\boldsymbol{h}}:\mathbb{Z}^2\to\mathbb{R}^C$ as:*

$$\widehat{\boldsymbol{h}}\{\boldsymbol{f}\}(\boldsymbol{x}) := \boldsymbol{f}(\boldsymbol{x})+\mu_{\boldsymbol{b}_m}(\boldsymbol{x}), \quad (3.17)$$

where:

$$\mu_{\boldsymbol{b}}(\boldsymbol{x}) = \Theta_{\boldsymbol{b}}\frac{\mathbb{E}\left[\boldsymbol{\zeta}(\boldsymbol{b})\begin{bmatrix}-\sin(\{\boldsymbol{\zeta}\circledast\boldsymbol{f}\})\\ \cos(\boldsymbol{\zeta}\circledast\boldsymbol{f})\end{bmatrix}^T\left\{w\star\begin{bmatrix}\cos(\boldsymbol{\zeta}\circledast\boldsymbol{f})\\ \sin(\boldsymbol{\zeta}\circledast\boldsymbol{f})\end{bmatrix}\right\}\right]}{\mathbb{E}\left[\begin{bmatrix}\cos(\boldsymbol{\zeta}\circledast\boldsymbol{f})\\ \sin(\boldsymbol{\zeta}\circledast\boldsymbol{f})\end{bmatrix}^T\left\{w\star\begin{bmatrix}\cos(\boldsymbol{\zeta}\circledast\boldsymbol{f})\\ \sin(\boldsymbol{\zeta}\circledast\boldsymbol{f})\end{bmatrix}\right\}\right]}. \quad (3.18)$$

Then $\widehat{\boldsymbol{h}}\{\boldsymbol{f}\}(\boldsymbol{x})$ is equivalent to $\boldsymbol{h}\{\boldsymbol{f}\}(\boldsymbol{x})$ in (2.4)*.*

Proof. Let $\boldsymbol{e}:\mathbb{Z}^2\to\mathbb{R}^{B^2C}$ be defined as (3.11). Define a new random vector

$$\boldsymbol{Z} := \begin{bmatrix}\boldsymbol{\zeta}(\boldsymbol{b}_1)\\ \vdots\\ \boldsymbol{\zeta}(\boldsymbol{b}_{B^2})\end{bmatrix}. \quad (3.19)$$

Or equivalently, $\boldsymbol{Z}\sim\mathcal{N}(0,\Theta^{-1})$, where $\Theta\in\mathbb{R}^{B^2C\times B^2C}$ is as defined in (3.12). Substituting Theorem 1 into Lemma 3 and recalling that $\boldsymbol{f}(\boldsymbol{x}-\boldsymbol{b}_m)=\boldsymbol{f}(\boldsymbol{x})$, we have

$$\begin{aligned}\boldsymbol{h}\{\boldsymbol{f}\}(\boldsymbol{x}) &= \widehat{\boldsymbol{g}}_{\boldsymbol{b}_m}\{\boldsymbol{e}\}(\boldsymbol{x})\\ &= \boldsymbol{f}(\boldsymbol{x})+\Theta_{\boldsymbol{b}_m}\frac{\mathbb{E}\left[\boldsymbol{\zeta}(\boldsymbol{b}_m)\begin{bmatrix}-\sin(\boldsymbol{Z}^T\boldsymbol{e}(\boldsymbol{x}))\\ \cos(\boldsymbol{Z}^T\boldsymbol{e}(\boldsymbol{x}))\end{bmatrix}^T\left\{w(\boldsymbol{x})\star\begin{bmatrix}\cos(\boldsymbol{Z}^T\boldsymbol{e}(\boldsymbol{x}))\\ \sin(\boldsymbol{Z}^T\boldsymbol{e}(\boldsymbol{x}))\end{bmatrix}\right\}\right]}{\mathbb{E}\left[\begin{bmatrix}\cos(\boldsymbol{Z}^T\boldsymbol{e}(\boldsymbol{x}))\\ \sin(\boldsymbol{Z}^T\boldsymbol{e}(\boldsymbol{x}))\end{bmatrix}^T\left\{w(\boldsymbol{x})\star\begin{bmatrix}\cos(\boldsymbol{Z}^T\boldsymbol{e}(\boldsymbol{x}))\\ \sin(\boldsymbol{Z}^T\boldsymbol{e}(\boldsymbol{x}))\end{bmatrix}\right\}\right]}. \quad (3.20)\end{aligned}$$

Furthermore, the inner product $\boldsymbol{Z}^T\boldsymbol{e}(\boldsymbol{x})$ can be rewritten as a convolution-sum in (2.8), as follows

$$\boldsymbol{Z}^T\boldsymbol{e}(\boldsymbol{x}) = \sum_{\boldsymbol{b}\in\Gamma} \boldsymbol{\zeta}^T(\boldsymbol{b})\boldsymbol{f}(\boldsymbol{x}-\boldsymbol{b}) = \{\boldsymbol{\zeta} \circledast \boldsymbol{f}\}(\boldsymbol{x}). \tag{3.21}$$

Substituting (3.21) into (3.20) proves the Theorem. □

Steps required to carry out Theorem 2 are summarized in Algorithm 2. Similar to FSBF, the expectation operator in FSNLM is approximated by the Monte-Carlo averaging over L random vectors. The convolution operator is shared between the denominator and all of the color channels in the numerator, reducing the number of convolutions (denoted by $\star$) to $\mathcal{O}(L)$. (Compare this to SNLM with $\mathcal{O}(CL)$ convolutions.) As described earlier, the complexity of convolution-sum defined in (2.8) is invariant to B and C, and comparable to a conventional convolution if implemented with fast Fourier Transform (FFT).

Algorithm 1 Fast Stochastic Bilateral Filter

input: $\boldsymbol{f} : \mathbb{Z}^2 \to \mathbb{R}^C$
output: $\widehat{\boldsymbol{g}} : \mathbb{Z}^2 \to \mathbb{R}^C$
parameters: σ, Θ
initialize numerator $\boldsymbol{n}(\boldsymbol{x}) \Leftarrow 0$ and denominator $d(\boldsymbol{x}) \Leftarrow 0$
for L times **do**
  generate $\boldsymbol{\zeta} \sim \mathcal{N}(\boldsymbol{0}, \Theta^{-1})$
  compute $\rho(\boldsymbol{x}) \Leftarrow \boldsymbol{\zeta}^T\boldsymbol{f}(\boldsymbol{x})$
  compute $\epsilon(\boldsymbol{x}) \Leftarrow \cos(\rho(\boldsymbol{x}))$ and $\lambda(\boldsymbol{x}) = \sin(\rho(\boldsymbol{x}))$
  compute $\gamma(\boldsymbol{x}) \Leftarrow w(\boldsymbol{x}) \star \epsilon(\boldsymbol{x})$ using [44]
  compute $\beta(\boldsymbol{x}) \Leftarrow w(\boldsymbol{x}) \star \lambda(\boldsymbol{x})$ using [44]
  update $\boldsymbol{n}(\boldsymbol{x}) \Leftarrow \boldsymbol{n}(\boldsymbol{x}) + \boldsymbol{\zeta}\,(\beta(\boldsymbol{x})\epsilon(\boldsymbol{x}) - \gamma(\boldsymbol{x})s(\boldsymbol{x}))$
  update $d(\boldsymbol{x}) \Leftarrow d(\boldsymbol{x}) + \epsilon(\boldsymbol{x})\gamma(\boldsymbol{x}) + \lambda(\boldsymbol{x})\beta(\boldsymbol{x})$
end for
set $\widehat{\boldsymbol{g}}(\boldsymbol{x}) \Leftarrow \boldsymbol{f}(\boldsymbol{x}) + \Theta\boldsymbol{n}(\boldsymbol{x})/d(\boldsymbol{x})$

Algorithm 2 Fast Stochastic Non-Local Means

input: $\boldsymbol{f} : \mathbb{Z}^2 \to \mathbb{R}^C$
output: $\widehat{\boldsymbol{h}} : \mathbb{Z}^2 \to \mathbb{R}^C$
parameter: σ, Θ
initialize numerator $\boldsymbol{n}(\boldsymbol{x}) \Leftarrow 0$
initialize denominator $d(\boldsymbol{x}) \Leftarrow 0$
for L times **do**
  generate $\boldsymbol{\zeta} \in \mathbb{R}^{B\times B\times C}$: $\boldsymbol{\zeta}(\boldsymbol{b}) \sim \mathcal{N}(0, \Theta_b^{-1})$, $\forall \boldsymbol{b} \in \boldsymbol{\Gamma}$
  compute $\rho(\boldsymbol{x}) \Leftarrow \boldsymbol{\zeta}(\boldsymbol{x}) \circledast \boldsymbol{f}(\boldsymbol{x})$
  compute $\epsilon(\boldsymbol{x}) \Leftarrow \cos(\rho(\boldsymbol{x}))$ and $\lambda(\boldsymbol{x}) = \sin(\rho(\boldsymbol{x}))$
  compute $\gamma(\boldsymbol{x}) \Leftarrow w(\boldsymbol{x}) \star \epsilon(\boldsymbol{x})$ using [44]
  compute $\beta(\boldsymbol{x}) \Leftarrow w(\boldsymbol{x}) \star \lambda(\boldsymbol{x})$ using [44]
  update $\boldsymbol{n}(\boldsymbol{x}) \Leftarrow \boldsymbol{n}(\boldsymbol{x}) + \boldsymbol{\zeta}(\boldsymbol{b}_m)\,(\beta(\boldsymbol{x})\epsilon(\boldsymbol{x}) - \gamma(\boldsymbol{x})\lambda(\boldsymbol{x}))$
  update $d(\boldsymbol{x}) \Leftarrow d(\boldsymbol{x}) + \epsilon(\boldsymbol{x})\gamma(\boldsymbol{x}) + \lambda(\boldsymbol{x})\beta(\boldsymbol{x})$
end for
set $\widehat{\boldsymbol{h}}(\boldsymbol{x}) \Leftarrow \boldsymbol{f}(\boldsymbol{x}) + \Theta_{\boldsymbol{b}_m} \boldsymbol{n}(\boldsymbol{x})/d(\boldsymbol{x})$

Algorithm 3 Fast Stochastic Block Non-Local Means

input: $\boldsymbol{f} : \mathbb{Z}^2 \to \mathbb{R}^C$
output: $\widehat{\boldsymbol{i}} : \mathbb{Z}^2 \to \mathbb{R}^C$
parameter: σ, Θ
initialize numerator $\boldsymbol{n_b}(\boldsymbol{x}) \Leftarrow 0$
initialize denominator $d(\boldsymbol{x}) \Leftarrow 0$
for L times **do**
  generate $\boldsymbol{\zeta} \in \mathbb{R}^{B\times B\times C}$: $\boldsymbol{\zeta}(\boldsymbol{b}) \sim \mathcal{N}(0, \Theta_b^{-1})$, $\forall \boldsymbol{b} \in \boldsymbol{\Gamma}$
  compute $\rho(\boldsymbol{x}) \Leftarrow \boldsymbol{\zeta}(\boldsymbol{x}) \circledast \boldsymbol{f}(\boldsymbol{x})$
  compute $\epsilon(\boldsymbol{x}) \Leftarrow \cos(\rho(\boldsymbol{x}))$ and $\lambda(\boldsymbol{x}) = \sin(\rho(\boldsymbol{x}))$
  compute $\gamma(\boldsymbol{x}) \Leftarrow w(\boldsymbol{x}) \star \epsilon(\boldsymbol{x})$ using [44]
  compute $\beta(\boldsymbol{x}) \Leftarrow w(\boldsymbol{x}) \star \lambda(\boldsymbol{x})$ using [44]
  update $\boldsymbol{n_b}(\boldsymbol{x}) \Leftarrow \boldsymbol{n_b}(\boldsymbol{x}) + \boldsymbol{\zeta}(\boldsymbol{b})\,(\beta(\boldsymbol{x})\epsilon(\boldsymbol{x}) - \gamma(\boldsymbol{x})\lambda(\boldsymbol{x}))$
  update $d(\boldsymbol{x}) \Leftarrow d(\boldsymbol{x}) + \epsilon(\boldsymbol{x})\gamma(\boldsymbol{x}) + \lambda(\boldsymbol{x})\beta(\boldsymbol{x})$
end for
set $\widehat{\boldsymbol{i}}(\boldsymbol{x}) \Leftarrow \boldsymbol{f}(\boldsymbol{x}) + B^{-2} \sum_{\boldsymbol{b}\in\Gamma} \Theta_{\boldsymbol{b}} \boldsymbol{n_b}(\boldsymbol{x}+\boldsymbol{b})/d(\boldsymbol{x})$

Table 3.1: Complexity of bilateral filtering implementations. The bottlenecks are shown in red. C=# color/spectrum of filtering image, W=window size, Q=# quantization steps, K=# summations in [2] or # clustering in [4], L=# Monte-Carlo draws. Expected per-pixel costs of convolution and clustering are of order $\mathcal{O}(1)$ and $\mathcal{O}(CKL)$, respectively.

	per pixel					per image	
	multiply	add	divide	exp/sin/cos	memory	convolution	clusters
Original BF [7–9]	$(2C+2)W^2$	$(2C-1)W^2 + (W^2-1)(C+1)$	C	W^2	$1+2C$	0	0
Paris [6]	$(C+2)Q^C$	$(C+1)Q^C$	C	Q^C	CQ^C	2	0
Chaudhury [2] Sugimoto [18]	$(3C+1)K^C$	$CK^C + 2K^{C-1}$	C	K^C	$1+2C$	$(C+1)K^C$	0
Deng [19]	$(3C+1)K^C$	$CK^C + 2K^{C-1} + C$	C	$2K^C$	$1+2C$	K^C	0
Karam [17]	$(5C+2)L$	$2CL + (C+1)(L-1)$	C	$2L$	$1+2C$	$(2C+2)L$	0
Sugimoto [3]	$(C+1)K$	$KC + 2(K-1)$	$C+1$	K	$1+2C$	$(C+1)K$	K
Nair [4]	CK	CK	C	K	$1+2C$	$(C+1)K$	K
FSBF (proposed)	$(2C+4)L$	$(C+1)L + (C+1)(L-1) + C$	C	$2L$	$1+2C$	$2L$	0

Table 3.2: Complexity analysis of non-local means implementations. Complexity bottleneck is marked in **red**. C=# color/spectrum of filtering image, W=window size, B=block size, P = percentage of pixels kept in a window, L=# Monte-Carlo draws. Expected per-pixel cost of convolution is of order $\mathcal{O}(1)$.

	per pixel						per image	
	multiply	add	divide	exp/sin/cos	compare	sqrt	conv/FFT	sqrt
Original NLM [15]	$(B^2C+2+C)W^2$	$(2B^2C-1)W^2$ $+(W^2-1)(C+1)$	C	W^2	0	0	0	0
BNLM	$(2B^2C+2)W^2+1$	$(2B^2C-1)W^2$ $+(W^2-1)(B^2C+1)$ $+(B^2-1)C$	CB^2	W^2	0	0	0	0
Dauwe [20]	$(B^2C+2+C)W^2P$ $+10$	$(2B^2C-C)W^2P+(C-1)$ $+(W^2P-1)(C+1)$ $+3W^2+24$	$C+1$	W^2P	$6W^2$	1	0	1
Chan [21]	$(B^2C+2+C)W^2P$	$(2B^2C-C)W^2P+(C-1)$ $+(W^2P-1)(C+1)$	C	W^2P	0	0	0	0
Goossens [22]	$CW^2+(C+1)W^2$	$(2C-1)W^2$ $+(C+1)(W^2-1)$	C	W^2	0	0	W^2	0
Karam [17]	$(5C+2)L$	$2CL+(C+1)(L-1)$	C	$2L$	0	0	$(2C+3)L$ $+C$	0
FSNLM (proposed)	$(C+4)L$	$2L+(C+1)(L-1)+C$	C	$2L$	0	0	$3L+C$	0
FSBNLM (proposed)	$(B^2C+4)L$	$2L+(B^2C+1)(L-1)$ $+B^2C$	CB^2	$2L$	0	0	$3L+C$	0

3.2 Complexity Analysis and Further Acceleration Techniques

3.2.1 Convergence Rate

The complexities of FSBF in Algorithm 1 and FSNLM in Algorithm 2 are dominated by the Gaussian filters. Recent advancements in filter designs established that Gaussian filtering can be approximated by $\mathcal{O}(1)$ per-pixel complexity processes using feedback [45–48] or short-time discrete cosine transform [43, 44, 49, 50]. Hence the per-pixel complexity of FSBF/FSNLM implemented with such filters would be $\mathcal{O}(L)$, i.e. invariant to the filter window size, image size.

Furthermore, we prove below that the iteration number L grows very slowly with the increased color dimension. Recalling Theorem 1, the convergence rate of Monte-Carlo averaging in FSBF and FSNLM is proportional to the variances of $\cos(\boldsymbol{\zeta}^T\boldsymbol{q})$ and $\boldsymbol{\zeta}\sin(\boldsymbol{\zeta}^T\boldsymbol{q})$ in Lemma (2).

Corollary 1. *Let* $\boldsymbol{\zeta} \sim \mathcal{N}(0, \Theta^{-1})$, $\boldsymbol{\zeta} \in \mathbb{R}^C$ *be a normal random vector, and* $\boldsymbol{q} \in \mathbb{R}^C$. *Then the variance of* $\cos(\boldsymbol{\zeta}^T \boldsymbol{q})$ *is no greater than* $\frac{1}{2}$. *The covariance matrix of* $\boldsymbol{\zeta} \sin(\boldsymbol{\zeta}^T \boldsymbol{q})$ *is*

$$\Theta^{-1} \frac{1 - \alpha^2}{2} + \Theta^{-1} \boldsymbol{q}\boldsymbol{q}^T \Theta^{-1} (2\alpha^2 - \alpha) \tag{3.22}$$

where

$$\alpha = \exp\left(-\boldsymbol{q}^T \Theta^{-1} \boldsymbol{q}\right). \tag{3.23}$$

The proof is found in Appendix A. The significance of Corollary 1 is that the Monte-Carlo convergence rate of $\phi(\boldsymbol{q}) = \mathbb{E}[\cos(\boldsymbol{\zeta}^T \boldsymbol{q})]$ in the denominator of (3.9) is bounded by a constant—i.e. invariant with respect to C (in the worst case; faster in the usual case). The numerator of (3.8) is similarly governed by $\Theta \nabla \phi(\boldsymbol{q}) = \mathbb{E}[\Theta \boldsymbol{\zeta} \sin(\boldsymbol{\zeta}^T \boldsymbol{q})]$ in (3.10), whose Monte-Carlo convergence rate is proportional to the diagonal entries of its covariance matrix

$$\Theta \frac{1 - \alpha^2}{2} + \boldsymbol{q}\boldsymbol{q}^T (2\alpha^2 - \alpha). \tag{3.24}$$

We therefore conclude that the overall convergence rate of FSBF is invariant to the color dimension C, and so MSE scales only with the number of convolutions (as determined by L only). See Figure 4.1. It ensures that the accuracy requirement for higher color dimensional images can be met without increasing the number of iterations significantly.

3.2.2 Acceleration By Quasi-Random Numbers

An additional approach to accelerating the proposed FSBF/FSNLM is to reduce the number L of random vectors drawn in Monte-Carlo itself. Specifically, we replace the random vectors by the so-called *quasi-random numbers*—low discrepancy sequences whose overall distribution is consistent with the random variables of interest. Owing to the fact that quasi-random numbers are designed to be more evenly distributed than genuine random

variables, various studies have confirmed that Monte-Carlo with quasi random numbers have more desirable convergence properties.

Let $\boldsymbol{\pi}_\ell \in \mathbb{R}^C$, $\ell \in \{1, \ldots, L\}$ be a Sobol sequence that fills the C-dimensional interval $(0,1)^C$ in a uniform manner [51]. Given a random vector $\boldsymbol{\zeta} \sim \mathcal{N}(0, \Theta)$, $\boldsymbol{\zeta} \in \mathbb{R}^C$ and any function $\Omega : \mathbb{R}^C \to \mathbb{R}$, the following approximation holds for a sufficiently large L:

$$\begin{aligned} E[\Omega(\boldsymbol{\zeta})] &= \int_{\mathbb{R}^C} \Omega(u)\,\mathrm{pdf}(u)du \\ &= \lim_{L\to\infty} \frac{1}{L}\sum_{\ell=1}^{L} \Omega\left(\mathrm{cdf}^{-1}(\boldsymbol{\pi}_\ell)\right) \approx \frac{1}{L}\sum_{\ell=1}^{L} \Omega\left(\mathrm{cdf}^{-1}(\boldsymbol{\pi}_\ell)\right), \end{aligned} \tag{3.25}$$

where the $\mathrm{pdf} : \mathbb{R}^C \to \mathbb{R}$ and $\mathrm{cdf} : \mathbb{R}^C \to \mathbb{R}$ are the probability and cumulative density functions of $\boldsymbol{\zeta}$, respectively. Using this property, we replace the Monte-Carlo random vector $\boldsymbol{\zeta}$ in Algorithms 1 and 2 with $\mathrm{cdf}^{-1}(\boldsymbol{\pi}_\ell)$. We refer to this implementation as the Quasi-FSBF (QFSBF) and Quasi-FSNLM (QFSNLM). Experiments in Section 4.1 confirm a faster convergence of QFSBF, but not for QFSNLM.

3.2.3 Fast Stochastic Block Non-Local Means

Levering block/patch-based denoising idea that pixels within the $B \times B$ block share filtering weights, suppose we approximate the conventional NLM in (2.4) by a "block non-local means" (BNLM) of the form:

$$\boldsymbol{i}\{\boldsymbol{f}\}(\boldsymbol{x}) := \sum_{\boldsymbol{b}\in\Gamma} \frac{\displaystyle\sum_{\boldsymbol{y}\in\mathbb{Z}^2} w(\boldsymbol{x}+\boldsymbol{b}-\boldsymbol{y})\cdot\psi\{\boldsymbol{f}\}(\boldsymbol{x}+\boldsymbol{b},\boldsymbol{y})\cdot\boldsymbol{f}(\boldsymbol{y}-\boldsymbol{b})}{B^2\displaystyle\sum_{\boldsymbol{y}\in\mathbb{Z}^2} w(\boldsymbol{x}+\boldsymbol{b}-\boldsymbol{y})\cdot\psi\{\boldsymbol{f}\}(\boldsymbol{x}+\boldsymbol{b},\boldsymbol{y})}. \tag{3.26}$$

Intuitively, we have allowed pixels within the block $\boldsymbol{y}-\boldsymbol{b}$ to be averaged together using the shifted weights $\psi\{\boldsymbol{f}\}(\boldsymbol{x},\boldsymbol{y})$—to compensate for the shift, we shift the output pixel in the opposite direction (hence $\boldsymbol{x}+\boldsymbol{b}$ term). This can be written equivalently as:

$$\boldsymbol{i}\{\boldsymbol{f}\}(\boldsymbol{x}) = \frac{1}{B^2}\sum_{\boldsymbol{b}\in\Gamma} \boldsymbol{g}_{\boldsymbol{b}}\{\boldsymbol{e}\}(\boldsymbol{x}+\boldsymbol{b}) \tag{3.27}$$

where $\boldsymbol{g_b}\{\boldsymbol{f}\}(\boldsymbol{x}+\boldsymbol{b})$ the shifted bilateral filter output in (3.15) for each pixel within the block. The BNLM complexity found in Table 3.2 is similar to that of the NLM, with some extra additions to carry out (3.27) and a small overhead for carrying out (3.15) B^2 times. In practice, BNLM approximates NLM well because they share the same weights $\psi\{\boldsymbol{f}\}(\boldsymbol{x},\boldsymbol{y})$. Averaged over 90 images, the mean squared error $\frac{1}{C}\mathbb{E}||\boldsymbol{h}-\boldsymbol{i}||^2$ between NLM and BNLM was only 11 when $B=3$.

Thanks to the averaging in (3.27), BNLM has a great potential for Monte-Carlo acceleration (despite the additional complexity relative to NLM). Fast stochastic version of BNLM follows directly from Theorem 2:

Corollary 2 (FSBNLM)**.** *Let* $\boldsymbol{\zeta}(\boldsymbol{b}) \sim \mathcal{N}(0,\Theta_{\boldsymbol{b}}^{-1})$*,* $\boldsymbol{\zeta}(\boldsymbol{b}) \in \mathbb{R}^C$*, be independent random vectors defined over* $\boldsymbol{b} \in \boldsymbol{\Gamma}$*. Define* $\widehat{\boldsymbol{i}} : \mathbb{Z}^2 \to \mathbb{R}^C$ *as:*

$$\begin{aligned}\widehat{\boldsymbol{i}}\{\boldsymbol{f}\} =& \frac{1}{B^2}\sum_{\boldsymbol{b}\in\Gamma}\widehat{\boldsymbol{g}}_{\boldsymbol{b}}\{\boldsymbol{e}\}(\boldsymbol{x}+\boldsymbol{b}) \\ =& \boldsymbol{f}(\boldsymbol{x}) + \frac{1}{B^2}\sum_{\boldsymbol{b}\in\Gamma}\mu_{\boldsymbol{b}}(\boldsymbol{x}+\boldsymbol{b})\end{aligned} \tag{3.28}$$

where $\mu_{\boldsymbol{b}}(\boldsymbol{x})$ *is as defined in* (3.18)*. Then* $\widehat{\boldsymbol{i}}\{\boldsymbol{f}\}(\boldsymbol{x})$ *is equivalent to* $\boldsymbol{i}\{\boldsymbol{f}\}(\boldsymbol{x})$ *in* (3.26)*.*

Proof is omitted because it is very similar to Theorem 2. In terms of complexity, (3.28) is comparable to (3.17)—the additional cost of computing $\mu_{\boldsymbol{b}}$ for all $\boldsymbol{b} \neq \boldsymbol{b}_m$ is almost negligible because $\mu_{\boldsymbol{b}}$ and $\mu_{\boldsymbol{b}_m}$ share the convolution operations. Yet, FSBNLM converges significantly faster than FSNLM. To see why this is the case, recall that the Monte-Carlo convergence rate of FSNLM is proportional to (3.24). The averaging in (3.28) reduces this covariance matrix to:

$$[\boldsymbol{I}_C/B^2,\ldots,\boldsymbol{I}_C/B^2]\left(\Theta\frac{1-\alpha^2}{2}+\boldsymbol{q}\boldsymbol{q}^T(2\alpha^2-\alpha)\right)\begin{bmatrix}\boldsymbol{I}_C/B^2\\ \vdots \\ \boldsymbol{I}_C/B^2\end{bmatrix} = \frac{1-\alpha^2}{B^4}\sum_{\boldsymbol{b}\in\Gamma}\Theta_{\boldsymbol{b}} + \frac{2\alpha^2-\alpha}{B^4}\sum_{\boldsymbol{b}\in\Gamma}z_{\boldsymbol{b}}, \tag{3.29}$$

where $\boldsymbol{I}_C \in \mathbb{R}^{C\times C}$ is an identity matrix and $\Theta_{\boldsymbol{b}} \in \mathbb{R}^{C\times C}$ is as defined in (3.12). This is effectively a reduction of covariance matrix by factor of B^2 (since the summation over $\boldsymbol{b} \in \Gamma$ takes B^2 elements). Hence we conclude that the Monte-Carlo convergence rate of FSBNLM is roughly B^2 times faster than $FSNLM$.

We experimentally verify the speedup in Figure 4.1(b). Here, the MSE is computed using the original NLM as a reference. The FSNLM converges to NLM faster than SNLM—while SNLM is slightly more computationally complex than FSNLM per iteration, the variance of (2.9) is more favorable than (3.24). By contrast, there is an implied MSE penalty associated with FSBNLM due to the fact that it converges to BNLM instead of NLM:

$$\mathbb{E}\|\boldsymbol{h} - \widehat{\boldsymbol{i}}\|^2 \geq \mathbb{E}\|\boldsymbol{h} - \boldsymbol{i}\|^2. \tag{3.30}$$

However, since FSBNLM converges B^2 times faster than FSNLM, the overall performance of FSBNLM is far more favorable.

CHAPTER IV

RESULTS: BILATERAL FILTER AND NON-LOCAL MEANS

4.1 Experimental Verification

Recall that in literature, NLM and BF serve different purposes—NLM is used as a denoising filter, while BF is used primarily for smoothing out textures while retaining the edges (often in computer vision). As such, the input data $\boldsymbol{f}$ in our experiments use non-noisy data for BF, while NLM experiments used noisy data (additive white Gaussian noise with $\sigma^2 = 25$).

Figure 4.1 shows the mean squared error convergence rate as a function of number of convolutions, averaged over 90 images, between the original and proposed implementations for the bilateral filter and the non-local means ($\frac{1}{C}\mathbb{E}\|\boldsymbol{g}-\widehat{\boldsymbol{g}}\|^2$, and $\frac{1}{C}\mathbb{E}\|\boldsymbol{h}-\widehat{\boldsymbol{h}}\|^2$ or $\frac{1}{C}\mathbb{E}\|\boldsymbol{h}-\widehat{\boldsymbol{i}}\|^2$), respectively. As expected FSBF requires far fewer convolutions to converge to desired outputs of bilateral filter as opposed to the SBF (Figure 4.1(a)). Similarly, as seen in Figure 4.1(b), FSNLM converges to NLM with fewer convolutions than the SNLM. It is also clear that the FSBNLM converges to the desired result with even fewer convolutions than both the SNLM and FSNLM implementations, with the MSE between BNLM and the original NLM being approximately 11 (averaged over 90 images).

The complexity analysis of Table 3.1 suggests the proposed bilateral filter implementation is four times faster than the SBF for color images (i.e when $C = 3$) as seen in the bottleneck of both algorithms. Similarly, FSNLM is two times faster than the SNLM implementation. For the BF experiments, $\theta = 50$ converges faster than $\theta = 20$, which is expected because of Corollary 1. NLM is typically used for denoising, and so the experiment was run

for one set of parameters that yield the best visual results. With $\theta = 70$, a block size of 3×3 and a window of 19×19.

Figure 4.2 compares the execution time of the conventional bilateral filter, the implementations in [2], [3], [4], [17] and the proposed FSBF, for varying color dimension C. In this experiment, the range parameter θ is varied based on the number of channels C, as $\theta = 30\sqrt{C}$. This ensures that the bilateral weights $\phi(\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y}))$ remain relatively constant despite the increasing C. The reported times confirm that the proposed FSBF filter's complexity grows very slowly with respect to C. The implementation of [3] runs out of memory at $C = 16$, with large number of clusters $K > 250$. We report the times it takes for the methods in [17], [3], [4], [2] to yield an MSE (averaged over color channels) of 5 or less. The implementations in [2], [4] and [3], albeit fast for grayscale and color, tend to be slower than the proposed method - and sometimes even the conventional BF - with the increase of color dimension. Their execution times grow at a faster-than-linear rate. These times are based on Matlab R2019a running on a 2016 ThinkStation P300, with Intel Xeon E3-1241 v3, 32GB RAM, and 1.5TB HDD, NVidia Quadro K620. We omit a similar test for non-local means because its original implementation of NLM cannot process the high-dimensional images in a reasonable time.

Figures 4.3 and 4.4 show the results of BF, BF implementation of [4], and QFSBF for a color input image ($C = 3$) and a hyperspectral image ($C = 31$), respectively. From the execution times, we can see that, for the color image, the [4] has a 21-times speedup over the conventional BF, whereas the QFSBF has a 66-fold speedup. In the case of the hyperspectral image, the speedup between the BF implementation of [4] and the BF is approximately 16 times, and the QFSBF further speeds it up by additional 3-fold. Figure 4.5 shows the result of the non-local means and block non-local means implementations for

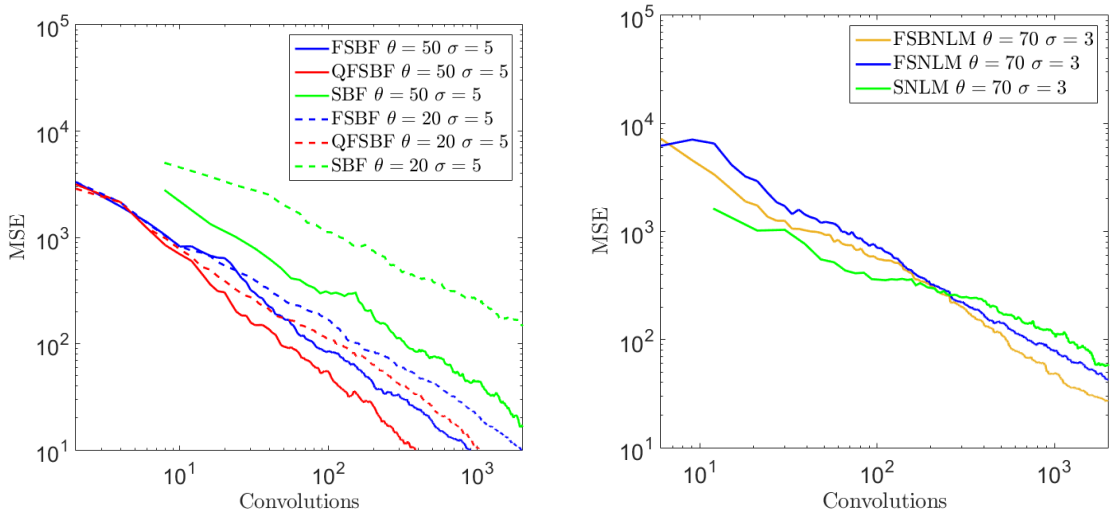


Figure 4.1: Graph showing the MSE (averaged over color channels) (a) for SBF, FSBF and QFSBF with respect to BF; (b) for SNLM and FSNLM with respect to NLM, and FSBNLM with respect to BNLM. The results are obtained averaging over 100 images taken from the McGill Color Image Database in [1].

denoising. The SNLM execution is 10 times faster than the original NLM, while FSNLM and FSBNLM speed ups are 29 and 6 times, respectively.

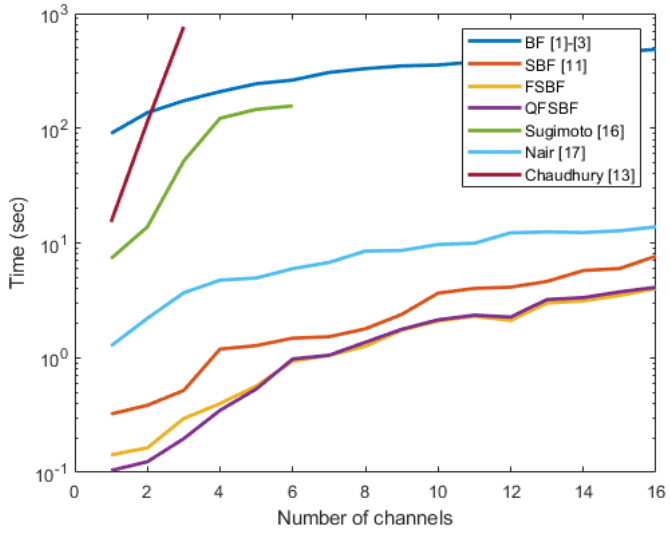
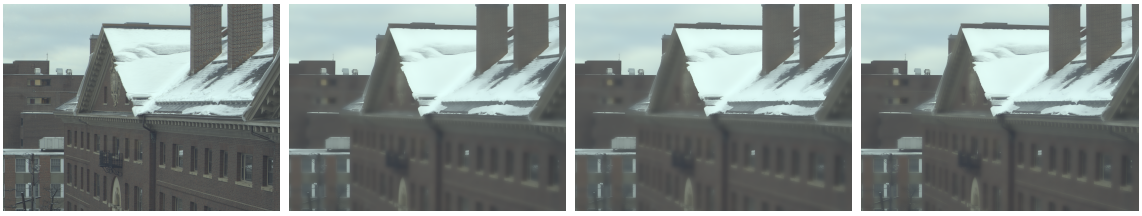


Figure 4.2: Graph showing the execution time (necessary to achieve MSE averaged over color channels of 5) of the BF, SBF, FSBF, QFSBF as well as the BF implementations of [2], [3], [4], as a function of color channels C. A hyperspectral image from set in [5] (1392 x 1040 x 31) was used.

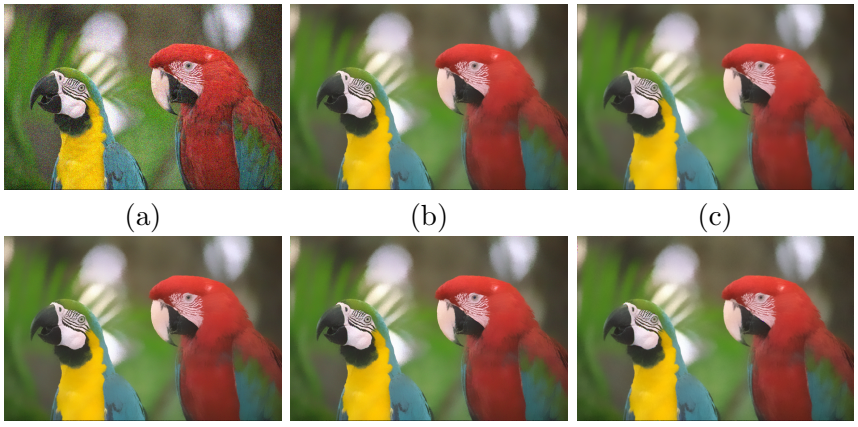
(a) (b) (c) (d)

Figure 4.3: Example bilateral filtering results. Parameters were (a-d), $\sigma = 10$, $\theta = 51$, $W = 61$.(a) Input image (b) BF (61.98 sec), (c) BF implementation of [4] (2.96 sec), (d) Quasi-FSBF (0.93 sec). Iteration numbers were chosen to yield PSNR ≥ 41.14 dB relative to the BF output. Image from [6].



(a) (b) (c) (d)

Figure 4.4: Example bilateral filtering results. Parameters were (a-d), $\sigma = 10$, $\theta = 130$, $W = 61$. (a) Input image (b) BF (1209.29 sec), (c) BF implementation of [4] (76.29 sec), (d) FSBF (62.43 sec). Iteration numbers were chosen to yield PSNR $\geq$ 41.14 dB relative to the BF output. Image from set in [5] (1392 x 1040 x 31)



(a) (b) (c)

(d) (e) (f)

Figure 4.5: Example non-local means results. Parameters used: $\sigma = 10$, $\theta = 85$, $W = 61$, $B = 3$. Execution times (b) NLM (635.902 sec, PSNR = 31.24 dB), (c) BNLM (804.997 sec, PSNR =30.50 dB), (d) SNLM (62.07 sec, PSNR = 30.08 dB), (e) FSNLM (21.85 sec, PSNR = 30.02 dB), (f) FSBNLM (134.49 sec, PSNR = 30.07 dB). Iteration numbers were chosen to yield a PSNR $\geq$ 30 relative to the clean image. Parrot image from Kodak color image set (768 x 512).

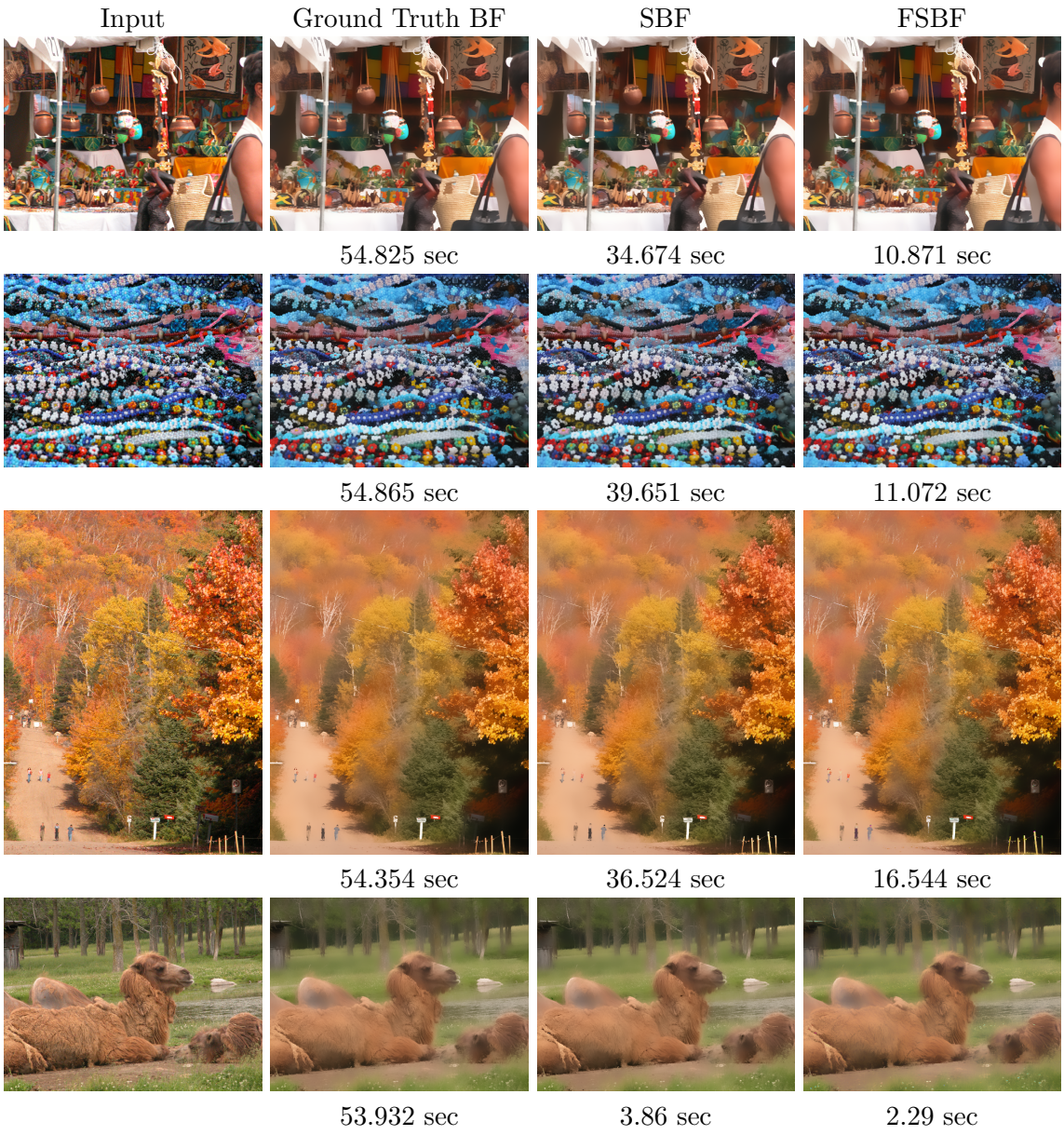


Figure 4.6: Example bilateral filtering results. Parameters were (a-d), $\sigma = 10$, $\theta = 130$, $W = 61$. Reported times are obtained to yield an MSE of 5 between BF and SBF, as well as BF and FSBF.

Noisy input | BNLM 912.204 sec | FSBNLM 415.587 sec

NLM 720.281 sec | SNLM 129.828 sec | FSNLM 89.827 sec

Figure 4.7: Example non-local means and block non-local means results. Parameters used: $\sigma = 10$, $\theta = 85$, $W = 61$, $B = 3$. Reported times are obtained to yield an MSE of 5 between naive implementations and fast versions.

Noisy input | BNLM 908.236 sec | FSBNLM 338.384 sec

NLM 724.926 sec | SNLM 147.747 sec | FSNLM 108.683 sec

Figure 4.8: Example non-local means and block non-local means results. Parameters used: $\sigma = 10$, $\theta = 85$, $W = 61$, $B = 3$. Reported times are obtained to yield an MSE of 5 between naive implementations and fast versions.

Noisy input | BNLM 892.918 sec | FSBNLM 283.233 sec

NLM 709.326 sec | SNLM 183.997 sec | FSNLM 93.229 sec

Figure 4.9: Example non-local means and block non-local means results. Parameters used: $\sigma = 10$, $\theta = 85$, $W = 61$, $B = 3$. Reported times are obtained to yield an MSE of 5 between naive implementations and fast versions.

Noisy input | BNLM 835.183 sec | FSBNLM 350.515 sec

NLM 652.691 sec | SNLM 202.263 sec | FSNLM 94.073 sec

Figure 4.10: Example non-local means and block non-local means results. Parameters used: $\sigma = 10$, $\theta = 85$, $W = 61$, $B = 3$. Reported times are obtained to yield an MSE of 5 between naive implementations and fast versions.

CHAPTER V

BACKGROUND: DISTANCE TRANSFORM

5.1 Background and Related Works

A "metric" $d : \mathbb{R}^L \times \mathbb{R}^L \to [0, \infty)$ is an L-dimensional distance function, where $d(\boldsymbol{x}, \boldsymbol{y})$ measures the distance between vectors $\boldsymbol{x} = (x_1, \ldots, x_L)^T \in \mathbb{R}^L$ and $\boldsymbol{y} = (y_1, \ldots, y_L)^T \in \mathbb{R}^L$. These include the p-norm (Minkowski distance):

$$d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|_p = \sqrt[p]{\sum_{\ell=1}^{L} |x_\ell - y_\ell|^p}. \tag{5.1}$$

The input to the distance function is a binary-valued image $F : \mathbb{Z}^L \to \{0, 1\}$, where we interpret "0" to be a background pixel and "1" to be the pixel of interest, which we refer to as an "edge." See Figure 5.1(a). Then the distance transform $D : \mathbb{Z}^L \to \mathbb{Z}$ in Figure 5.1(b) is a gray-scale image defined as

$$D(\boldsymbol{x}) = \min_{\boldsymbol{y}: F(\boldsymbol{y})=1} d(\boldsymbol{x}, \boldsymbol{y}). \tag{5.2}$$

(a) $F(\boldsymbol{x})$ (b) $D(\boldsymbol{x})$ (c) Algorithm 1 (d) Algorithm 2 (f) Algorithm 3

Figure 5.1: (a) Input image. (b) Ground truth distance transform (Euclidean norm) by brute-forced implementation. (c) Log-sum-conv approximation in Theorem 3. (d) Soft minimum approximation in Theorem 4. (e) Deriv-log-conv approximation in Theorem 5. Used parameters $\lambda = 0.35$ and $\Delta\lambda = 0.01$.

Hence the value $D(\boldsymbol{x})$ corresponds to the minimum distance between pixel location ($\boldsymbol{x} \in \mathbb{Z}^L$) and the surrounding edge pixels ($\boldsymbol{y} \in \mathbb{Z}^L$ where $F(\boldsymbol{y}) = 1$). Carrying out (5.2) involves up to N comparisons for each $\boldsymbol{x}$, or $\mathcal{O}(N^2)$ complexity overall.

Techniques for reducing the complexity of distance transforms fall into the following categories: *propagation*, *raster-scanner*, and *separable scanning*. Propagation algorithms compute the distance transform in (5.2) by progressively moving away from the edge pixels ($F(\boldsymbol{y}) = 1$), recording the distances along the way [52–57]. Raster-scanning algorithms approximate the Euclidean distance by Chamfer distance, using local masks chosen to minimize the approximation error [28, 29, 58]. The complexity-accuracy trade-offs of 4-neighborhood and 8-neighborhood masks have been studied thoroughly [25].

Separable scanning algorithms reduce (5.2) into a series of L independent one-dimensional operations by tracking parabola intersections [59–62] or by morphology operators [63–65]. They have $\mathcal{O}(N^{1.5})$ worst-case performance but $\mathcal{O}(N)$ complexity in special cases. The Voronoi diagram intersections has $\mathcal{O}(N \log N)$ complexity, which reduces to $\mathcal{O}(N)$ by exploiting the intersection of a image row/column and the Voronoi diagram seed pixels [66–69]. Voronoi diagram implementation is cumbersome, however. We have summarized the complexity of various algorithm in the supplementary material.

CHAPTER VI

PROPOSED METHOD: FAST CONVOLUTIONAL DISTANCE TRANSFORMS

6.1 Proposed: Convolutional Distance Transform

6.1.1 Minimum Functions

Distance transform in (5.2) is computationally expensive because of the "minimum" function. The minimum function is highly nonlinear, making it difficult to accelerate. We review three alternative forms of minimum functions—smooth approximation to minimum functions—used frequently in modern machine learning algorithms. When substituted into the definition of distance transform in (5.2), the algorithm can be approximated efficiently using convolution operators.

We rewrite minimum function as log-sum-exponential.

Lemma 4 (Log-Sum-Exp)**.** *Let* $z_1, \ldots, z_K \in \mathbb{R}$. *Then:*

$$\min\{z_1, \ldots, z_K\} = \lim_{\lambda \to 0} -\lambda \log \left(\sum_{k=1}^{K} \exp\left(-\frac{z_k}{\lambda}\right) \right). \tag{6.1}$$

Proof. Without the loss of generality, assume

$$z_1 = \cdots = z_{K_0} < z_{K_0+1} \leq \cdots \leq z_K. \tag{6.2}$$

Here, $K_0 = 1$ implies unique minimum (i.e. $z_1 < z_2$). Thus,

$$-\lambda \log \left(\sum_{k=1}^{K} \exp\left(-\frac{z_k}{\lambda}\right) \right) = \lambda \log \left(K_0 \exp\left(-\frac{z_1}{\lambda}\right) + \sum_{k=K_0+1}^{K} \exp\left(-\frac{z_k}{\lambda}\right) \right). \tag{6.3}$$

Using $\log(a+b) = \log(a) + \log(1+b/a)$, (6.3) becomes:

$$\begin{aligned} &- \lambda \log \left(K_0 \exp\left(-\frac{z_1}{\lambda}\right) \right) - \lambda \log \left(1 + \frac{\sum_{k=K_0+1}^{K} \exp\left(-\frac{z_k}{\lambda}\right)}{K_0 \exp\left(-\frac{z_1}{\lambda}\right)} \right) \\ &= -\lambda \log K_0 + z_1 - \lambda \log \left(1 + \frac{1}{K_0} \sum_{k=K_0+1}^{K} \exp\left(-\frac{z_k - z_1}{\lambda}\right) \right). \end{aligned}$$

Taking the limit as $\lambda \to 0$, it converges to z_1. □

Lemma 5 is another approximation to a minimum function.

Lemma 5 (Soft Minimum)**.** *Let $z_1, \ldots, z_K \in \mathbb{R}$. Then:*

$$\min\{z_1, z_2, \ldots, z_K\} = \lim_{\lambda \to 0} \frac{\sum_{k=1}^{K} z_k \exp\left(-\frac{z_k}{\lambda}\right)}{\sum_{k=1}^{K} \exp\left(-\frac{z_k}{\lambda}\right)}. \tag{6.4}$$

Proof. Assume (6.2), as before. Then

$$\begin{aligned} \lim_{\lambda \to 0} \frac{\sum_{k=1}^{K} z_k \exp\left(-\frac{z_k}{\lambda}\right)}{\sum_{k=1}^{K} \exp\left(-\frac{z_k}{\lambda}\right)} \left(\frac{\exp\left(\frac{z_1}{\lambda}\right)}{\exp\left(\frac{z_1}{\lambda}\right)}\right) &= \lim_{\lambda \to 0} \frac{K_0 z_1 + \sum_{k=K_0+1}^{K} z_k \exp\left(-\frac{z_k - z_1}{\lambda}\right)}{K_0 + \sum_{k=K_0+1}^{K} \exp\left(-\frac{z_k - z_1}{\lambda}\right)} \\ &= \frac{K_0 z_1}{K_0} = z_1. \end{aligned} \tag{6.5}$$

□

The Lemmas 4 and 5 are exact *in limit*. For a practical implementation, we approximate using a small $\lambda > 0$ value:

$$\begin{aligned} \min\{z_1, z_2, \ldots, z_K\} &\approx -\lambda \log\left(\sum_{k=1}^{K} \exp\left(-\frac{z_k}{\lambda}\right)\right) \\ \min\{z_1, z_2, \ldots, z_K\} &\approx \frac{\sum_{k=1}^{K} z_k \exp\left(-\frac{z_k}{\lambda}\right)}{\sum_{k=1}^{K} \exp\left(-\frac{z_k}{\lambda}\right)}. \end{aligned} \tag{6.6}$$

The subsequent Theorems and Corollaries we develop below are similarly valid in limit $\lambda \to 0$. Thus it is understood that the approximations hold for a small $\lambda > 0$.

We may also extend Lemma 5 by the use of derivatives.

Corollary 3 (Deriv-Log-Sum-Exp)**.** *Let $z_1, \ldots, z_K \in \mathbb{R}$ be a set of real numbers. Then:*

$$\min\{z_1, \ldots, z_K\} = \lim_{\lambda \to 0} \lambda^2 \frac{\partial}{\partial \lambda} \log\left(\sum_{k=1}^{K} \exp\left(-\frac{z_k}{\lambda}\right)\right). \tag{6.7}$$

Proof. Recall $\frac{\partial}{\partial\lambda}\exp\left(-\frac{z_k}{\lambda}\right) = \frac{z_k}{\lambda^2}\exp\left(-\frac{z_k}{\lambda}\right)$. Then by substituting into Lemma 5,

$$\min\{z_1,\ldots,z_K\} = \lim_{\lambda\to 0}\lambda^2\frac{\frac{\partial}{\partial\lambda}\sum_{k=1}^{K}\exp\left(-\frac{z_k}{\lambda}\right)}{\sum_{k=1}^{K}\exp\left(-\frac{z_k}{\lambda}\right)}. \tag{6.8}$$

By derivative chain rule, (6.7) and (6.8) are equivalent. □

6.1.2 Algorithm 1: Log-Conv Approximation

In this dissertation, we restrict our attention to the "translation invariant" metrics. That is, $d(\cdot,\cdot)$ satisfies the property:

$$d(\boldsymbol{x},\boldsymbol{y}) = d(\boldsymbol{x}+\boldsymbol{z},\boldsymbol{y}+\boldsymbol{z}), \qquad \forall\boldsymbol{z}\in\mathbb{R}^L. \tag{6.9}$$

Setting $\boldsymbol{z} = -\boldsymbol{y}$, we see that $d(\boldsymbol{x},\boldsymbol{y})$ is a function of $\boldsymbol{x}-\boldsymbol{y}$:

$$d(\boldsymbol{x},\boldsymbol{y}) = d(\boldsymbol{x}-\boldsymbol{y},0) \tag{6.10}$$

Acknowledging slight abuse of notation, we henceforth use $d(\boldsymbol{x},\boldsymbol{y})$ and $d(\boldsymbol{x}-\boldsymbol{y})$ interchangeably. Translation invariant metrics include Euclidean distances and p-norms in (5.1).

Substituting Lemma 4 into the definition of distance transform allows us to rewrites (5.2) in terms of convolution.

Theorem 3 (Log-Conv)**.** *Let $\star$ denotes the convolution. Then*

$$D(\boldsymbol{x}) = \lim_{\lambda\to 0} -\lambda\log\left(F(\boldsymbol{x})\star\exp\left(-\frac{d(\boldsymbol{x})}{\lambda}\right)\right). \tag{6.11}$$

Proof. Substitute Lemma 4 into the distance transform in (5.2):

$$\begin{aligned} D(\boldsymbol{x}) &= \lim_{\lambda\to 0} -\lambda\log\left(\sum_{\boldsymbol{y}:F(\boldsymbol{y})=1}\exp\left(-\frac{d(\boldsymbol{x},\boldsymbol{y})}{\lambda}\right)\right) \\ &= \lim_{\lambda\to 0} -\lambda\log\left(\sum_{\boldsymbol{y}\in\mathbb{Z}^L}F(\boldsymbol{y})\exp\left(-\frac{d(\boldsymbol{x}-\boldsymbol{y})}{\lambda}\right)\right). \end{aligned} \tag{6.12}$$

By the definition of convolution, Theorem is proved. □

Consider a p-norm distance metric. Then we may rewrite the distance transform as a set of separable convolutions.

Corollary 4 (Separable Log-Conv)**.** *Let $d(\cdot,\cdot)$ denote a p-norm distance metric in* (5.1)*. Define $\overset{\ell}{\star}$ as a one-dimensional convolution in the ℓ-th dimension. Then*

$$D(\boldsymbol{x}) = \lim_{\lambda\to 0} \sqrt[p]{-\lambda \log\left(F(\boldsymbol{x}) \overset{1}{\star} \exp\left(-\frac{x_1^p}{\lambda}\right) \overset{2}{\star} \ldots \overset{L}{\star} \exp\left(-\frac{x_L^p}{\lambda}\right)\right)}. \tag{6.13}$$

Proof. It follows directly from (6.11) that

$$D(\boldsymbol{x}) = \lim_{\lambda\to 0} \sqrt[p]{-\lambda \log\left(F(\boldsymbol{x}) \star \exp\left(-\frac{d(\boldsymbol{x})^p}{\lambda}\right)\right)}. \tag{6.14}$$

By the definition of the p-norm, we have

$$\exp\left(-\frac{d(\boldsymbol{x})^p}{\lambda}\right) = \prod_{k=1}^{K} \exp\left(-\frac{x_k^p}{\lambda}\right). \tag{6.15}$$

Substituting (6.15), the convolution in (6.14) is separable. □

6.1.3 Algorithm 2: Soft Minimum Approximation

Similarly, we can approximate the distance transform by leveraging the soft minimum approximation in Lemma 5.

Theorem 4 (Soft Minimum Approximation)**.**

$$D(\boldsymbol{x}) = \lim_{\lambda\to 0} \frac{F(\boldsymbol{x}) \star \left(d(\boldsymbol{x}) \exp\left(-\frac{d(\boldsymbol{x})}{\lambda}\right)\right)}{F(\boldsymbol{x}) \star \exp\left(-\frac{d(\boldsymbol{x})}{\lambda}\right)}. \tag{6.16}$$

Proof. Substituting Lemma 5 into (5.2), we have:

$$\begin{aligned} D(\boldsymbol{x}) &= \lim_{\lambda\to 0} \frac{\sum_{\boldsymbol{y}:F(\boldsymbol{y})=1} d(\boldsymbol{x},\boldsymbol{y}) \exp\left(-\frac{d(\boldsymbol{x},\boldsymbol{y})}{\lambda}\right)}{\sum_{\boldsymbol{y}:F(\boldsymbol{y})=1} \exp\left(-\frac{d(\boldsymbol{x},\boldsymbol{y})}{\lambda}\right)} \\ &= \lim_{\lambda\to 0} \frac{\sum_{\boldsymbol{y}\in\mathbb{Z}^L} F(\boldsymbol{y}) d(\boldsymbol{x}-\boldsymbol{y}) \exp\left(-\frac{d(\boldsymbol{x}-\boldsymbol{y})}{\lambda}\right)}{\sum_{\boldsymbol{y}\in\mathbb{Z}^L} F(\boldsymbol{y}) \exp\left(-\frac{d(\boldsymbol{x}-\boldsymbol{y})}{\lambda}\right)}. \end{aligned} \tag{6.17}$$

By definition of convolution, Theorem is proved. □

6.1.4 Algorithm 3: Deriv-Log-Conv Approximation

We rewrite the distance transform based on Corollary 3.

Theorem 5 (Deriv-Log-Conv Approximation)**.**

$$D(\boldsymbol{x}) = \lim_{\lambda \to 0} \lambda^2 \frac{\partial}{\partial \lambda} \log \left(F(\boldsymbol{x}) \star \exp\left(-\frac{d(\boldsymbol{x})}{\lambda} \right) \right). \tag{6.18}$$

Proof. Substituting Corollary 3 into (5.2), we have

$$\begin{aligned} D(\boldsymbol{x}) &= \lim_{\lambda \to 0} \lambda^2 \frac{\partial}{\partial \lambda} \log \left(\sum_{\boldsymbol{y}: F(\boldsymbol{y})} \exp\left(-\frac{d(\boldsymbol{x}, \boldsymbol{y})}{\lambda} \right) \right) \\ &= \lim_{\lambda \to 0} \lambda^2 \frac{\partial}{\partial \lambda} \log \left(\sum_{\boldsymbol{y} \in \mathbb{Z}^L} F(\boldsymbol{y}) \exp\left(-\frac{d(\boldsymbol{x} - \boldsymbol{y})}{\lambda} \right) \right). \end{aligned} \tag{6.19}$$

By the definition of convolution, Theorem is proved. □

Recall the definition of derivative:

$$\frac{\partial}{\partial \lambda} f(\lambda) = \lim_{\Delta\lambda \to 0} \frac{f(\lambda + \Delta\lambda) - f(\lambda)}{\Delta\lambda}. \tag{6.20}$$

Hence for a small $\Delta\lambda$, we have the practical approximation:

$$D(\boldsymbol{x}) \approx \frac{\lambda^2}{\Delta\lambda} \log \left(F(\boldsymbol{x}) \star \exp\left(-\frac{d(\boldsymbol{x})}{\lambda + \Delta\lambda} \right) \right) - \frac{\lambda^2}{\Delta\lambda} \log \left(F(\boldsymbol{x}) \star \exp\left(-\frac{d(\boldsymbol{x})}{\lambda} \right) \right). \tag{6.21}$$

In the special case that distance function is a p-norm in (5.1), we have the following separable implementation.

Corollary 5 (Separable Deriv-Log-Conv)**.** *Let $d(\cdot, \cdot)$ denote a p-norm distance metric in* (5.1)*. Then*

$$D(\boldsymbol{x}) = \lim_{\lambda \to 0} \sqrt[p]{\lambda^2 \frac{\partial}{\partial \lambda} \log \left(F(\boldsymbol{x}) \overset{1}{\star} \exp\left(-\frac{x_1^p}{\lambda} \right) \overset{2}{\star} \ldots \overset{L}{\star} \exp\left(-\frac{x_L^p}{\lambda} \right) \right)}. \tag{6.22}$$

Proof. Recall $\frac{\partial}{\partial\lambda}\exp\left(-\frac{d(x)^p}{\lambda}\right) = \frac{d(x)^p}{\lambda^2}\exp\left(-\frac{d(x)^p}{\lambda}\right)$. So,

$$\begin{aligned}&\lim_{\lambda\to 0}\lambda^2\frac{\partial}{\partial\lambda}\log\left(F(\boldsymbol{x})\star\exp\left(-\frac{d(\boldsymbol{x})^p}{\lambda}\right)\right) \\ &= \lim_{\lambda\to 0}\lambda^2\frac{F(x)\star\left(\frac{\partial}{\partial\lambda}\exp\left(-\frac{d(x)^p}{\lambda}\right)\right)}{F(x)\star\exp\left(-\frac{d(x)^p}{\lambda}\right)} = \min_{y:F(y)=1} d(x,y)^p.\end{aligned} \tag{6.23}$$

Taking the p-th root makes (6.23) a distance transform. By (6.15), the convolution in (6.23) becomes separable. □

6.2 Discussions

6.2.1 Complexity Analysis And Implementation Issues

The significance of the Theorems 3-5 is that distance transform in (5.2) is rewritten using convolution operators. Hence its complexity reduces to $\mathcal{O}(N\log N)$ when it is implemented using fast Fourier transform (FFT). Corollaries 4 and 5 extend Theorems 3 and 5 to separable convolution filters, respectively. The complexity of the brute-forced separable filter implementation reduces to $\mathcal{O}(N^{1.5})$—a speed-up compared to the brute-forced implementation of Theorem 3 with $\mathcal{O}(N^2)$. When implemented using FFT, however, there is no advantage to separable filtering—the complexity of N dimensional separable convolutional filtering remains at $\mathcal{O}(N\log N)$. Nevertheless, separable filtering is desirable in applications where only small distances $d(\boldsymbol{x},\boldsymbol{y})$ may be of interest. We may threshold the metric function as follows:

$$d'(\boldsymbol{x},\boldsymbol{y}) = \begin{cases} d(\boldsymbol{x},\boldsymbol{y}) & \text{if } |x_\ell - y_\ell| < \tau,\ \forall \ell \in \{1,\ldots,L\} \\ \infty & \text{else.}\end{cases} \tag{6.24}$$

The corresponding convolution filter has a finite impulse response (FIR): $\exp(-d'(\boldsymbol{x})^p/\lambda) = 0$ when if $|x_\ell - y_\ell| > \tau$. The complexity of the separable FIR filtering reduces to $\mathcal{O}(\tau N)$. Contrast this to the $\mathcal{O}(\tau^L N)$ complexity of non-separable FIR filtering. Moreover, in the

special case that $p = 2$, Corollary 3 is a separable Gaussian filter. Recently, the so-called "constant" one-dimensional Gaussian filter approximations have been proposed [70]. Using such Gaussian filtering reduces the overall complexity to $\mathcal{O}(N)$.

The exponential terms $\exp(-d^p/\lambda)$ in Corollaries 4 and 5 decays faster than $\exp(-d/\lambda)$ in Theorems 3-5. Even the double precision floating point may not enough to support $d(\cdot)^p$ if maximum $d(\cdot)$ is large (i.e $\exp(-d^p/\lambda)$ maps to zero), unless λ is larger. We overcame this problem by multiplexing multiple λ values. For example, we may replace (6.13) as

$$D(\boldsymbol{x}) \approx \left(\min \left(-\lambda \log \left(F(\boldsymbol{x}) \overset{1}{\star} \exp\left(-\frac{x_1^p}{\lambda} \overset{2}{\star} \dots \right) \right), -\lambda' \log \left(F(\boldsymbol{x}) \overset{1}{\star} \exp\left(-\frac{x_1^p}{\lambda'} \overset{2}{\star} \dots \right) \right) \right) \right)^{1/p}, \tag{6.25}$$

where $\lambda < \lambda'$. It is straightforward to update (6.18) using a similar strategy. Although the multiplexing strategy in (6.25) doubles the computational complexity, the separable filtering implementations in Corollaries 4 and 5 are considerably faster.

6.2.2 Contributions

There are several overall advantages to the proposed distance transform relative the prior art. First, the implementation is straightforward. Unlike the Voronoi diagram methods in [66–69], implementation in a high level language (such as Matlab) requires only a few lines of code, and the code is essentially identical for any signal dimensionality L. Second, the acceleration techniques in Theorems 3-5 are agnostic to the distance metric $d : \mathbb{Z}^L \times \mathbb{Z}^L \to \mathbb{R}$—we achieved $\mathcal{O}(N \log N)$ complexity for *any* translation-invariant distance metrics in general (i.e. not limited to p-norm or Euclidean distance functions). For example, it is popular to use *robust* distance metrics that are non-symmetric (e.g. orientation-sensitive) or non-polynomial (e.g. thresholded distance) in some applications. See Figure 7.1. In such cases, the proposed implementation of distance transform is preferred. Third,

at $\mathcal{O}(N)$ complexity, our method is competitive with the state-of-the-art distance transform implementations designed specifically for Euclidean distance transforms. However, the proposed approach comes at the cost of the approximation error (albeit negligible when λ value is small; see Section 7.1).

CHAPTER VII

RESULTS: FAST CONVOLUTIONAL DISTANCE TRANSFORM

7.1 Experimental Results

We implemented the Convolutional Distance Transform Algorithms 1-3 introduced in Section 6.1, using parameters $\lambda = 0.35$ and $\Delta\lambda = 0.01$. They are chosen based on the machine precision of Matlab. In particular, the smallest number in the $\exp(\cdot)$ function that still yields a non-trivial number is -745. We assumed that the largest distance d_{max} would be half the image size. Therefore we chose parameters to satisfy $d_{max}/2/\lambda = 745$ ($d_{max} = 512$ in our experiments). As evidenced by the results in Figure 7.3 and the approximation errors (stemming from small λ values) reported in Table 7.1, the output images from Algorithms 1-3 and their separable versions are practically indistinguishable from the brute-forced implementation (i.e. ground truth). The cross section of the distance transform in Figure 7.2 confirms that smaller λ and $\Delta\lambda$ values yield more accurate results. More images are available in the supplementary material.

Table 7.1: Mean squared error (MSE) of the approximated distance transform. Results averaged over twenty 512× 512 images.

	Algorithm 1		Algorithm 2	Algorithm 3	
	Theorem 3	Corollary 4	Theorem 4	Theorem 5	Corollary5
MSE	0.101	0.156	0.045	0.153	0.029

(a) (b)

Figure 7.1: Example illustrating various distance metrics. (a) Robust distance transform using $d(\boldsymbol{x}, \boldsymbol{y}) = min(20, \|\boldsymbol{x} - \boldsymbol{y}\|)$. (b) Non-symmetric distance transform (directionally biased) using $d(\boldsymbol{x}, \boldsymbol{y}) = \|[1, 0.8; 0.8, 1](\boldsymbol{x} - \boldsymbol{y})\|$. Implemented with Algorithm 1 with $\lambda = 0.35$.

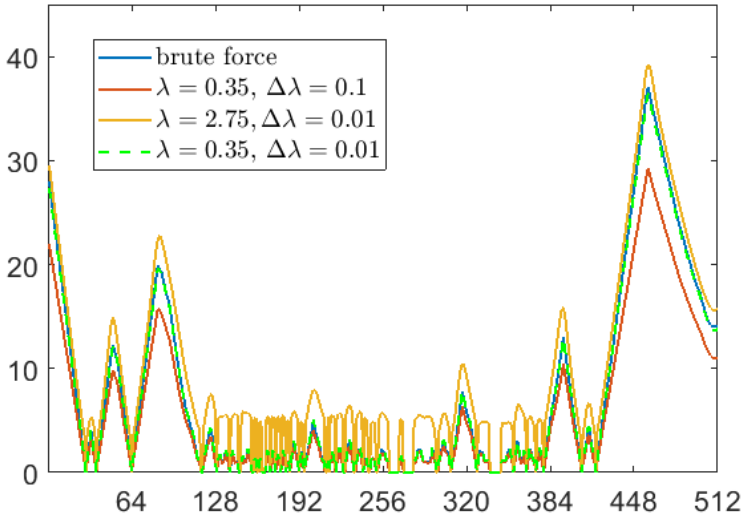
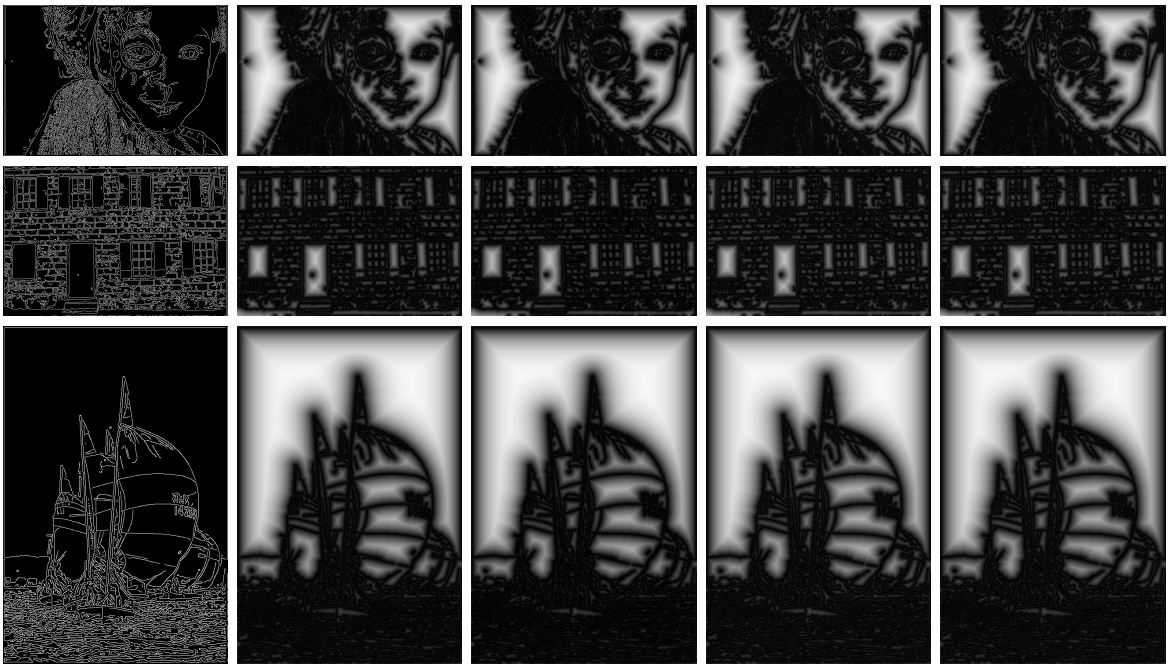


Figure 7.2: Plot illustrating the cross-section of the brute-force distance transform, and the implementation of Theorem 5 with different parameter values. With $\lambda = 0.35$ and $\Delta\lambda = 0.01$, the convolutional distance transform is indistinguishable from the brute force (ground truth) distance transform. Theorems 3 and 4 behave similarly.



(a) $F(\boldsymbol{x})$ (b) $D(\boldsymbol{x})$ (c) Algorithm 1 (d) Algorithm 2 (f) Algorithm 3

Figure 7.3: (a) Input image. (b) Ground truth distance transform (Euclidean norm) by brute-forced implementation. (c) Log-sum-conv approximation in Theorem 1. (d) Soft minimum approximation in Theorem 2. (e) Deriv-log-conv approximation in Theorem 3. Used parameters $\lambda = 0.35$ and $\Delta\lambda = 0.01$.

Table 7.2: Complexity of various distance transform implementations.

	Complexity	Order
Brute Force	$\mathcal{O}(N^2)$	ℓ_1, ℓ_2, ℓ_{inf}
[21]	$\mathcal{O}(N^{1.5})$	ℓ_p
[22]	$\mathcal{O}(N^{1.5})$	ℓ_2
[23]	$\mathcal{O}(N^{1.5})$	ℓ_p
[24]	$\mathcal{O}(N^{1.5})$	ℓ_2
[25]	$\mathcal{O}(N^{1.5})$	ℓ_2
[26]	$\mathcal{O}(N^{1.5})$	ℓ_2
[27]	$\mathcal{O}(N^{1.5})$	ℓ_2
[28]	$\mathcal{O}(N \log N)$	ℓ_1, ℓ_2
[29]	$\mathcal{O}(N \log N)$	ℓ_2
[30]	$\mathcal{O}(N \log N)$	ℓ_1, ℓ_2
[31]	$\mathcal{O}(N \log N)$	ℓ_2
Theorems 1-3	$\mathcal{O}(N \log N)$	ℓ_p, any translation invariant distance
Corollary 2-3	$\mathcal{O}(N \log N)$	ℓ_p

CHAPTER VIII

BACKGROUND: MULTI-FRAME IMAGE DENOISING

In the last decade, the quality of smartphones cameras has improved dramatically. They are accessible, easily portable and so common that they have replaced the point-and-shoot cameras, all the while increasing the consumer awareness of image and video quality. The demand for high resolution cameras in smartphones has increased. Lower light efficient of each pixel sensor is a direct consequence of the demand of higher resolution cameras because the signal-to-noise ratio of the image sensor scales linearly with the pixel sensor size, and smaller pixels suffer from increased noise. Therefore image and video denoising play a very important role in the quality of smartphone cameras.

Many image denoising algorithms have been developed over the years. Nonlinear spatial filtering [15], iterative methods for cost minimization [71] and directional filtering [72] aim to preserve edges while smoothing out the noise [73]. These methods have yielded satisfactory results in medical imaging where texture is not always critical. However, they do not do well with natural images. Image-patch based processing is an alternative, where the spatial information within a window is used to leverage the structure of neighboring pixel intensity patterns through sparse dictionary [74] or inter-patch similarity [15]. Other popular approaches include wavelet [75], discrete cosine transform [76], and principal component analysis [77]. The denoising performance of the aforementioned methods are limited by the signal-to-noise ratio of a single input frame. In low light conditions, these denoised images seem too smooth because it is hard to reconstruct the high spatial frequency details. By increasing the exposure, and thus the integration time, the signal-to-noise ratio while improve, but with it, the risks of image blur also increase which can result in the loss of content details.

Video denoising [78] can be considered to be a type of multi-frame image denoising having multiple frames instead of a single frame. However, their objectives are different: the quality of the video frames is evaluated collectively. A video has to look natural, and so for large motions, spatial details can be ignored, because the human eye cannot interpret it. In contrast, multi-frame image denoising aims at preserving the fine features such as edges, as well as texture, regardless of the motion in between frames. For that reason, the target exposure time of each frame is typically shorter than a video frame (1/200 to 1/100 seconds per frame as opposed to 1/60 to 1/30 seconds per frame) since the goal is to minimize blur. As a result, the frames are considerably noisier.

The idea of combining image sensor information along with IMU has proven useful in various applications such as image distortion caused by rolling shutters [79], image deblurring [80–83], multi-frame image deblurring [84], video stabilization [79] and generating panoramic images from video sequences [85]. Rolling shutters refers to the raster scan order in which image sensors capture the image. If the camera is in motion while capturing the image, different regions of the image are captured at slightly different times causing image distortions. In this case, IMUs are used to reconstruct the camera motion and determine the warping needed to correct the image. For image deblurring, the estimated blur kernel used is obtained from the IMU measurements which give a rough estimate of the camera motion. Said motion is used for non-blind image deblurring that require prior knowledge of the blur kernel. Recently, multi-shot image denoising has gained popularity [86], but as far as we are aware, our approach is novel. It relies on the Noise2Noise approach for denoising, and IMU-based deblurring and optical flow for image registration.

8.1 Image Denoising

Noise2Noise is a method that allows us to obtain clean images while only having access to noisy images [40, 41], irrespective of the noise model. It has proven useful by training neural networks to learn the statistical model of noise corruption indirectly from the data, as well as in MRI reconstructions from undersampled data [42]. It's also been useful in applications using pairs of noisy images, such as astrophotography, where typically, a long exposure is preferred. Albeit proven useful in practice, the claim presented in [42] has never been theoretically supported.

The Noise2Noise idea was developped based on the intuition that one can infer clean images based solely on looking at corrupted data. Using a set of unreliable data, estimating the true desired result relies on a loss function specific to the application in question. Minimizing the deviation between the result and the set of measurements, on average, is common practice to solving a problem of this nature, and is the basis of training neural networks.

In [42], they trained a "RED30" residual network [87] with known noise distributions, in particular: Gaussian, Poisson and Bernoulli noise, and compared the results to different denoising algorithms. Training the neural network was done twice: once using known clean images, and another, using only corrupted images. The results obtained showed that using clean images was unnecessary for this application, and that the two approaches yielded similar results.

Other applications in [42] include text removal. This experiments consists of images with strings of varying lengths, font colors and sizes placed randomly on the image, as well as on top of each other. The only difference as opposed to the noise removal, in this case,

is the loss function used during the training process. The results using clean targets for training, compared to that using only corrupted images show little to no difference.

Random-valued impulse noise removal was another success in using solely noisy data for training. This approach consists of replacing some pixels in the image with noise, rather than random salt and pepper noise. Again, in this case the loss function is the only difference in the training, and the results are on par with training using clean data.

The applications of Noise2Noise are many, and based on the experimental results in [42], the intuition that using only corrupted data to yield clean data seems solid. However, one can not test for all different noise distributions and all applications. For this reason, having a theoretical proof of concept is important to support the idea behind Noise2Noise.

8.2 Image Deblurring

This dissertation extends the work of IMU-based image deblurring described in [88]. (See Figure 8.1). There are three causes to blurry images: defocus blur, motion blur, and camera shake blur. As the name suggests, the defocus blur is when the object of the image is not in focus. The motion blur, is when the object of interest itself is in motion, whereas the camera shake blur is when the device used to capture the image has moved during long exposure times. The work in [88] is specific to the camera shake blur.

The camera shake blur is caused by an unknown motion of the device during the exposure time. That motion can be modeled by a translation vector ρ, and a rotation vector ψ represented by the blur arrows, and red arrows of Figure 8.2, respectively. These translations and rotation vectors can be modeled as a combination of the motion around the three axes

Defocus Blur | Motion Blur | Camera Shake

Figure 8.1: Types of blurs

x, y and z as such: $\rho = [\rho_{\mathbf{x}}, \rho_{\mathbf{y}}, \rho_{\mathbf{z}}]^{\mathbf{T}}$, and $\psi = [\psi_{\mathbf{x}}, \psi_{\mathbf{y}}, \psi_{\mathbf{z}}]^{\mathbf{T}}$.

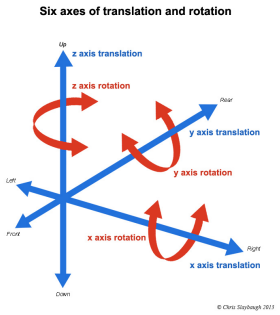


Figure 8.2: Motion vectors[16]

Each camera has a set of intrinsics parameter, which would allow to relate the points between the real world and its 2D projection on the camera sensor by a homography matrix. The camera intrinsics are given by:

$$\kappa = \begin{bmatrix} f & \gamma & c_{x0} \\ 0 & f & c_{y0} \\ 0 & 0 & 1 \end{bmatrix}, \tag{8.1}$$

where f is the focal length, c_0 the principal point and γ the skew parameter. These parameters are usually part of the hardware description, and the γ is set to zero if the image sensor pixels are square. The real world scene and its 2D projection on the camera sensor are related by the following homography matrix:

$$H(\rho, \psi) = \kappa \left\{ R(\psi) + \frac{\rho N^T}{d} \right\} \kappa^{-1} \tag{8.2}$$

where $R(\psi)$ is a rotation matrix combining the rotation around all the axes, and the term $\frac{\rho N^T}{d}$ is the translation term that can be approximated to zero if d, the distance of the scene to the camera, is large enough.

At each time instance of the exposure, the camera changes position, so the camera rotation has to be recorded for each of these time instances:

$$\theta(t) = [\theta_x(t), \theta_y(t), \theta_z(t)] \,. \tag{8.3}$$

Combining all of this together, we can relate the sharp image to the blurry image by the following equation:

$$g(n) = \frac{1}{T} \int_0^T f\left(H(\theta(t))n\right) dt + \epsilon(n), \tag{8.4}$$

where g, the blurry image, is the sharp image f transformed by the homography matrix $H(\theta(t))$ integrated over the exposure time $t \in [0, T]$ at each two-dimensional spatial location n.

There are three main ideas for image deblurring:

- Blind Deblurring, where solving two fixed-points iterations to get the blur kernel and the sharp image converge:

$$\left\{ \hat{f}, \hat{k} \right\} = \arg\min \|g - P\{f, k\}\|_2^2 + \alpha E_1(f) + \beta E_2(k) \tag{8.5}$$

- Non-Blind Deblurring, where the blur kernel is known, leaving only the recovery of the sharp image:

$$\hat{f} = \arg\min \|g - P\{f,k\}\|_2^2 + \alpha E_1(f) \tag{8.6}$$

- IMU-Deblurring, where the IMU data is present, but not to be completely trusted [88]:

$$\{\hat{f},\hat{\theta}\} = \arg\min \underbrace{\|g - \mathbb{Q}(f,\theta)\|_2^2}_{\text{image fidelity}} + \underbrace{\lambda\|\theta - \theta_0\|_p}_{\text{IMU fidelity}} + \alpha \underbrace{E_1(f)}_{\text{regularizer}}, \tag{8.7}$$

$$\{\hat{f},\hat{k}\} = \arg\min \underbrace{\|g - \mathbb{P}(f,k_\theta)\|_2^2}_{\text{image fidelity}} + \underbrace{\lambda\|k_\theta - k_{\theta_0}\|_q}_{\text{IMU fidelity}} + \alpha \underbrace{E_1(f)}_{\text{regularizer}} \tag{8.8}$$

where λ and α are parameters; g is the observed blurry image; θ_0 is the camera trajectory based on IMU data; f is the latent sharp image and θ is the latent camera motion; $E(f)$ is the regularizer that constrains the solution space of f; and $\mathbb{Q}(f,\theta)$ is the predicted blurry image g corresponding to the sharp image f and camera motion θ. The IMU-fidelity term can be replaced by a distance transform, in order to minimize the non-convex function:

$$\|\theta - \theta_0\|_p \approx T\|\mathbb{D}_p(\psi,\theta_0)k_\theta(\psi)\|_1. \tag{8.9}$$

The deblurring method mentioned above is unusual because it has two fidelity terms: image fidelity, and IMU fidelity to make sure that the deblurred image is consistent with both sensor datas. Unlike other methods, it also takes into account IMUs errors and does not consider it to be the ground truth. The drift errors are compensated by the image content, and inversely, the image motion is compensated by the IMU motion, until both motions agree.

CHAPTER IX

PROPOSED METHOD: MULTI-FRAME IMAGE DENOISING

The goal of this research is to develop a multi-frame image denoising technique for smartphones that leverages information from the camera sensor as well as the IMU. We take a set of N short exposure images, in low light, that will therefore be noisy. We denote these noisy images as $\{f_1, f_2, \ldots, f_N\}$. We then need to register the N images onto the same coordinate system before combining to obtain one denoised image. For that, we need to obtain a representation of the displacement in between frames, and although it can be inferred from the images themselves, it is not accurate in low light. For that reason, we acquire IMU data θ_0 that gives us a rough estimate of the trajectory, despite suffering from drift errors.

To perform the multi-frame image denoising task, we extend the deblurring method technique in (8.7). We propose to find the solution to the following minimization problem:

$$\{\widehat{f}, \widehat{\theta}\} = \arg\min_{f,\theta} \underbrace{\sum_{i=1}^{N} \|g_i - \mathbb{Q}(f, \theta_i)\|^2}_{\text{image fidelity}} + \underbrace{\lambda \sum_{i=1}^{N} \|\theta_i - \theta_{0,i}\|^2}_{\text{IMU fidelity}} + \underbrace{\alpha E(f)}_{\text{regularizer}} . \tag{9.1}$$

where $\widehat{\theta} = \{\widehat{\theta}_1, \ldots, \widehat{\theta}_N\}$. Comparing to a conventional single-frame image denoising technique of the form [89]:

$$\widehat{f} = \arg\min \|g - f\|^2 + \alpha E(f). \tag{9.2}$$

The multi-frame denoising in (9.1) incorporates N image fidelity terms that includes the registration step $\mathbb{Q}(f, \theta_i)$ where camera positions $\{\theta_1, \ldots, \theta_N\}$ need to be estimated. In contrast to the deblurring method in (8.7), there are N IMU fidelity terms that further constrain the solution space of the registration parameters. The added IMU fidelity term

determines how much confidence we have in the inertial measurements.

The proposed denoising equation in (9.1) can be re-written as:

$$\{\widehat{f},\widehat{H}\} = \arg\min_{f,H} \underbrace{\sum_{t=1}^{N} \|g(x,y,t) - f\left(H(t)\begin{bmatrix}x\\y\\1\end{bmatrix}\right)\|^2}_{\text{image fidelity}} + \underbrace{\lambda \sum_{t=1}^{N} \|H(t) - H_{IMU}(t)\|^2}_{\text{IMU fidelity}} + \underbrace{\alpha E(f)}_{\text{regularizer}}. \tag{9.3}$$

Here, the latent image f uses the homography matrix H to be registered to the noisy image g. This is done using the Taylor Series expansion in optical flow method as follows:

$$\begin{aligned} f\left(H(t)\begin{bmatrix}x\\y\\1\end{bmatrix}\right) &= f(x,y) + \left(\begin{bmatrix}1 & 0 & 0\end{bmatrix} H(t) \begin{bmatrix}x\\y\\1\end{bmatrix} - x\right)\frac{\partial}{\partial x} f(x,y) \\ &\quad + \left(\begin{bmatrix}0 & 1 & 0\end{bmatrix} H(t) \begin{bmatrix}x\\y\\1\end{bmatrix} - y\right)\frac{\partial}{\partial y} f(x,y) \\ &= f(x,y) + \left(\begin{bmatrix}h_x & s_x & t_x\end{bmatrix} \begin{bmatrix}x\\y\\1\end{bmatrix} - x\right)\frac{\partial}{\partial x} f(x,y) \\ &\quad + \left(\begin{bmatrix}h_y & s_y & t_y\end{bmatrix} \begin{bmatrix}x\\y\\1\end{bmatrix} - y\right)\frac{\partial}{\partial y} f(x,y). \end{aligned} \tag{9.4}$$

The homography matrix is formed of 6 terms as seen in the following equation:

$$H(t) = \begin{bmatrix} h_x & s_x & t_x \\ s_y & h_y & t_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{9.5}$$

where s represents the scaling coefficient in each direction, t the translation in x and y directions. The full derivation to obtain H can be found in Appendix A. The denoising problem comes down to finding the correct homography matrix H, as well as the latent clean image f. For this purpose, (9.3) can be split into two equations: one to find the ideal image, and the other to find the ideal homography. This is done by fixed point iteration until both results converge. The two equations in questions are as follows.

$$\widehat{f} = \arg\min_{f} \underbrace{\sum_{t=1}^{N} \|g(x,y,t) - f\left(\widehat{H}(t)\begin{bmatrix}x\\y\\1\end{bmatrix}\right)\|^2}_{\text{image fidelity}} + \underbrace{\alpha E(f)}_{\text{regularizer}} \tag{9.6}$$

$$\widehat{H} = \arg\min_{H} \underbrace{\sum_{t=1}^{N} \|g(x,y,t) - \widehat{f}\left(H(t)\begin{bmatrix}x\\y\\1\end{bmatrix}\right)\|^2}_{\text{image fidelity}} + \underbrace{\lambda \sum_{t=1}^{N} \|H(t) - H_{IMU}(t)\|^2}_{\text{IMU fidelity}} \tag{9.7}$$

Similarly to the deblurring method proposed by [88], replacing the IMU fidelity by a distance transform is feasible. It goes to say that the IMU data is somewhat accurate, but due to drift and noise issues, may not be exact. For that reason, we look for the best value within a certain distance of the location given to us by the gyroscope data.

Solving these equations comes down to splitting the problems into three parts, and tackling them separately before combining the results. We need to find the closest displacement from IMU data that corresponds to the image shift, register the images to the same reference image, then denoise the image.

As previously mentioned, the camera sensor and IMU are not synchronized, so finding the IMU timestamp corresponding to the image of interest is important before attempting image registration. To find the correct timestamp which yields the closest homography matrix to the real homography, we use the alternating direction method of multipliers (ADMM) which allows us to solves convex optimization problems by breaking them into smaller problems [90].

$$\widehat{H} = \arg\min_{H} \underbrace{\sum_{t=1}^{N} \|g(x,y,t) - \widehat{f}\left(H(t)\begin{bmatrix}x\\y\\1\end{bmatrix}\right)\|^2}_{\text{image fidelity}} + \underbrace{\lambda \sum_{t=1}^{N} \|H(t) - H_{IMU}(t_0)\|^2}_{\text{IMU fidelity}} \tag{9.8}$$

$$\widehat{t_0}(t) = \arg\min_{t_0} \|H(t) - H_{IMU}(t_0)\|^2 \tag{9.9}$$

Finding the closest IMU sample to the actual homography then allows us to regularize the homography obtained through the image fidelity term, and obtain $\widehat{H}$. This derivation can be found in Appendix B.

Once the images are registered, we can denoise them using the Noise2Noise approach. Obtaining the best denoising result is equivalent to getting as close to the clean image as possible. However, in the absence of a clean reference, the only accessible data are the noisy images. Theorems 6–11 prove the theory behind the Noise2Noise claims of [42].

Theorem 6 (Noise2Noise2Var)**.** *Let Y_i be noisy images, and X the clean image. Then,*

$$\sigma^2_{noise_1} = Var(Y_1) - Cov(Y_1, Y_2). \tag{9.10}$$

Proof. Expanding the variance and covariance and rewriting them in terms of X, shows that the two terms are equal.

$$\begin{aligned} Cov(Y_1, Y_2) &= \mathbb{E}[Y_1 Y_2] - \mathbb{E}[Y_1]\mathbb{E}[Y_2] \\ &= \mathbb{E}[X^2] - \big[\mathbb{E}[X]\big]^2 \\ &= Var(X). \end{aligned} \tag{9.11}$$

Rewriting $Cov(Y_1, Y_2)$ as $Var(X)$ is valid because of the following:

$$\begin{aligned} \mathbb{E}[Y_1] &= \mathbb{E}\big[\mathbb{E}[Y_1|X]\big] = \mathbb{E}[X] \\ \mathbb{E}[Y_1 Y_2] &= \mathbb{E}\big[\mathbb{E}[Y_1 Y_2|X]\big] \\ &= \mathbb{E}\big[\mathbb{E}[Y_1|X]\mathbb{E}[Y_2|X]\big] \\ &= \mathbb{E}[X^2]. \end{aligned} \tag{9.12}$$

Lastly, the total variance theorem gives:

$$\begin{aligned} Var(Y_k) &= Var\left(\mathbb{E}[Y_k|X]\right) + \mathbb{E}\big[Var(Y_k|X)\big] \\ &= \sigma^2_X + \sigma^2_{noise_k}, \end{aligned} \tag{9.13}$$

which, combined with (9.11), allows us to rewrite the noise variance σ^2_{noise} as (9.10). □

The goal of signal denoising is to get as close as possible to the clean signal. This can be expressed as a mean squared error, where we compare the clean image X to the denoised image $\phi(Y_i)$. Theorem 7 proves that the mean squared error can be rewritten solely in terms of Y_i.

Theorem 7 (Noise2Noise2MSE)**.** *Let $\phi(Y_i)$ be the denoised result of the corrupted input Y_i. In this case:*

$$\begin{aligned} MSE &= \mathbb{E}\big[(\phi(Y_1) - X)^2\big] \\ &= \mathbb{E}\big[(Y_2 - \phi(Y_1))^2\big] - \mathbb{E}[Y_2^2] + \mathbb{E}[Y_1 Y_2] \end{aligned} \tag{9.14}$$

Proof. Rewriting the terms containing the clean image X in terms of only the corrupted images Y_i, we get:

$$\begin{aligned} &\mathbb{E}\big[(\phi(Y_1) - Y_2)^2\big] - \mathbb{E}[Y_2^2] + \mathbb{E}[Y_1 Y_2] \\ =&\mathbb{E}[(\phi(Y_1))^2] + \mathbb{E}[Y_1 Y_2] - 2\mathbb{E}[\phi(Y_1) Y_2] \\ +&\mathbb{E}[Y_2^2] - \mathbb{E}[Y_2^2] \end{aligned} \tag{9.15}$$

Using (9.12), we get:

$$\begin{aligned} &\mathbb{E}\big[(\phi(Y_1) - Y_2)^2\big] - \mathbb{E}[Y_2^2] + \mathbb{E}[Y_1 Y_2] \\ =&\mathbb{E}[(\phi(Y_1))^2] + \mathbb{E}[X^2] - 2\mathbb{E}[\phi(Y_1) X] \\ =&\mathbb{E}\big[(\phi(Y_1) - X)^2\big], \end{aligned} \tag{9.16}$$

and writing the last term of (9.15) as a function of X yields:

$$\begin{aligned} \mathbb{E}[\phi(Y_1) Y_2] &= \mathbb{E}\big[\mathbb{E}[\phi(Y_1) Y_2 | X]\big] \\ &= \mathbb{E}\Big[\mathbb{E}[\phi(Y_1)|X]\mathbb{E}[Y_2|X]\Big] \\ &= \mathbb{E}[\phi(Y_1) X]. \end{aligned} \tag{9.17}$$

Combining the above equations proves the theorem. □

From Theorem 7, we are able to prove the claim presented in [42].

Corollary 6. *Let the denoising function $\phi(\cdot)$ be a parametric denoising function with parameter θ. Then, we can rewrite* (9.14) *as:*

$$\arg\min_\theta \mathbb{E}\big[\, (X - \phi_\theta(Y_1))^2 \,\big] = \arg\min_\theta \mathbb{E}\big[\, (Y_2 - \phi_\theta(Y_1))^2 \,\big] \tag{9.18}$$

By definition, the SSIM is a full reference metric. However, Theorem 8 allows us to rewrite it in terms of only noisy images, since the clean image X is inaccessible.

Theorem 8 (Noise2Noise2SSIM)**.** *The SSIM index is calculated to measure the similarity between X and $\phi(Y)$ as such:*

$$\begin{aligned} SSIM &= \frac{2\mathbb{E}[X]\mathbb{E}[\phi(Y_1)]}{(\mathbb{E}[X])^2 + (\mathbb{E}[\phi(Y_1)])^2} \times \frac{2Cov(X, \phi(Y_1))}{Var(X) + Var(\phi(Y_1))} \\ &= \frac{2\mathbb{E}[Y_2]\mathbb{E}[\phi(Y_1)]}{(\mathbb{E}[Y_2])^2 + (\mathbb{E}[\phi(Y_1)])^2} \times \frac{2Cov(Y_2, \phi(Y_1))}{Cov(Y_1, Y_2) + Var(\phi(Y_1))} \end{aligned} \tag{9.19}$$

Proof. Using (9.12), as well as:

$$\begin{aligned} Cov(Y_2, \phi(Y_1)) &= \mathbb{E}\big[\mathbb{E}[Y_2\phi(Y_1)|X]\big] - \mathbb{E}[Y_2]\mathbb{E}[\phi(Y_1)] \\ &= \mathbb{E}\big[\mathbb{E}[X\phi(Y_1)|X]\big] - \mathbb{E}[X]\mathbb{E}[\phi(Y_1)] \\ &= Cov(X, \phi(Y_1)), \end{aligned} \tag{9.20}$$

and replacing them in (9.19) proves the theorem. □

Theorem 9 (Noise2Noise2WSSIM)**.** *Using the wavelet domain equivalent of SSIM:*

$$WSSIM = \frac{2\mathbb{E}(Y_2\phi(Y_1))}{\mathbb{E}[Y_1Y_2] + \mathbb{E}[\phi(Y_1)^2]}. \tag{9.21}$$

Proof. By definition:

$$WSSIM = \frac{2\mathbb{E}[X\phi(Y_1)]}{\mathbb{E}[X^2] + \mathbb{E}[\phi(Y_1)^2]}, \tag{9.22}$$

where $\mathbb{E}[X] = \mathbb{E}[Y_i] = 0$. Using (9.12) and (9.17), and replacing them in (9.22) proves the theorem. □

$$\begin{aligned} WSSIM|Y_1 &= \frac{2\mathbb{E}[X\phi(Y_1)|Y_1]}{\mathbb{E}[X^2|Y_1] + \mathbb{E}[\phi(Y_1)^2|Y_1]} \\ &= \frac{2\phi(Y_1)\mathbb{E}[Y_2|Y_1]}{\mathbb{E}[X^2|Y_1] + \phi(Y_1)^2} \end{aligned} \tag{9.23}$$

Minimizing the mean squared error in 7 would give the closest denoised image $\phi(Y)$ to the clean image X.

Theorem 10 (Noise2Noise2MMSE)**.** *The MMSE result as described in* (9.24) *uses one noisy image Y_2 to denoise Y_1 as follows:*

$$\phi(Y_1 = y_0) = \mathbb{E}[Y_2|Y_1 = y_0] \tag{9.24}$$

Proof. Finding the minimum consists of taking the derivative of the MSE and setting it to zero:

$$\begin{aligned}\frac{\partial MSE}{\partial \phi(Y_1)} &= 2\phi(Y_1 = y_0) - 2\mathbb{E}\big[Y_2 \frac{\partial}{\partial \phi(Y_1 = y_0)}\phi(Y_1)|Y_1 = y_0\big] \\ &= 2\phi(Y_1 = y_0) - 2\mathbb{E}[Y_2|Y_1 = y_0] \\ &= 0.\end{aligned} \tag{9.25}$$

□

Similary, maximizing the structural similarity between the clean image X and noisy image Y_1 would give the closest denoised image to the clean image.

Theorem 11 (Noise2Noise2SSIMdenoise)**.** *Using two noisy images Y_2 and Y_3 would allow us to obtain $\phi(Y_1)$:*

$$\phi(Y_1) = sign(Y_1)\sqrt{\mathbb{E}[X^2|Y_1]} \tag{9.26}$$

Proof. Taking the derivative of the conditional WSSIM and setting it to zero would maximize the SSIM:

$$\begin{aligned}&\frac{\partial}{\partial \phi(Y_1)} WWSIM|Y_1 \\ &= \frac{2\mathbb{E}[Y_2|Y_1]}{\mathbb{E}[X^2|Y_1] + \phi(Y_1)^2} - \frac{4\phi(Y_1)^2\mathbb{E}[Y_2|Y_1]}{(\mathbb{E}[X^2|Y_1] + \phi(Y_1)^2)^2}\end{aligned} \tag{9.27}$$

where

$$\mathbb{E}[X^2|Y_1] = \phi(Y_1)^2 \tag{9.28}$$

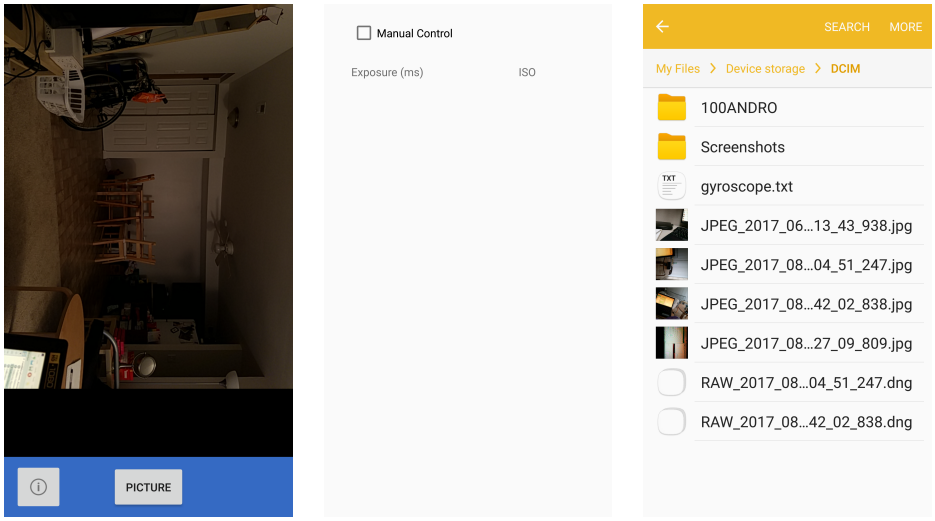
□

Combining all the steps would allow us to obtain one clean image, from a set of short exposure images.

CHAPTER X

RESULTS: MULTI-FRAME IMAGE DENOISING

In order for this method to work, we first need a smartphone application that allows us to record all the information needed for our denoising application. In particular, we need to be able to record and save the IMU data (gyroscope and accelerometer), the raw sensor data, and we need the ability to take a burst of several images, with a set exposure time for each one of them. Figure 10.1 shows the Android application and its settings page.



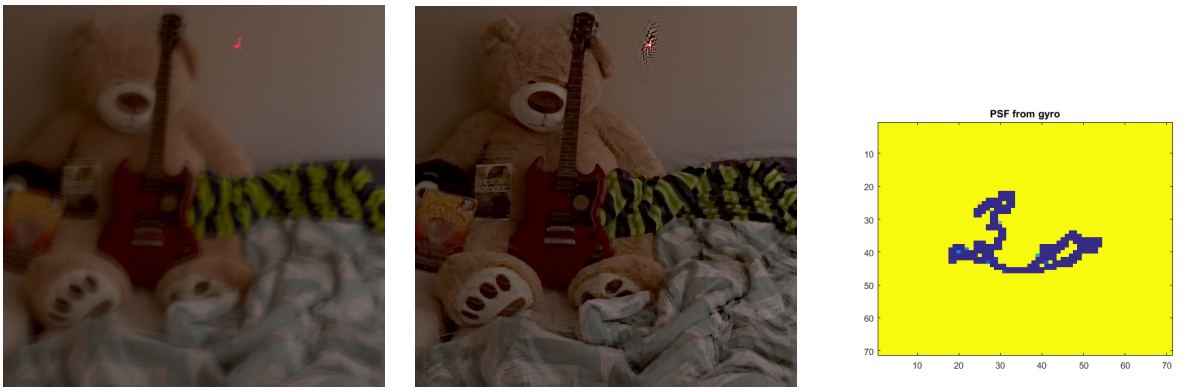
My Camera App Settings of Camera Camera recordings

Figure 10.1: Android camera application

10.1 Deblurring

The first step is to implement the deblurring algorithm proposed in [88] on mobile data. The following images show the results for different exposure times. In Figure 10.2, it is

clear that the gyroscope blur kernel is different from the ground truth kernel of the image, given to us by a laser pointer. This shows the synchronization problem between camera and IMU. Despite that synchronization, the deblurred image gives a satisfactory result.



Blurry input | Deblurred | Blur kernel from gyro data

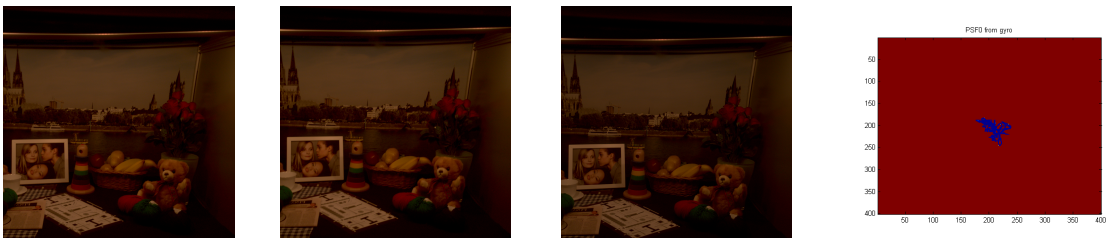
Figure 10.2: Results: 1000ms exposure

Blurry input | Deblurred

Figure 10.3: Results: 250ms exposure

Blurry input Deblurred

Figure 10.4: Results: 500ms exposure



(a) Second image y_2 (b) Fifth image y_5 (c) Seventh image y_7 (d) IMU data

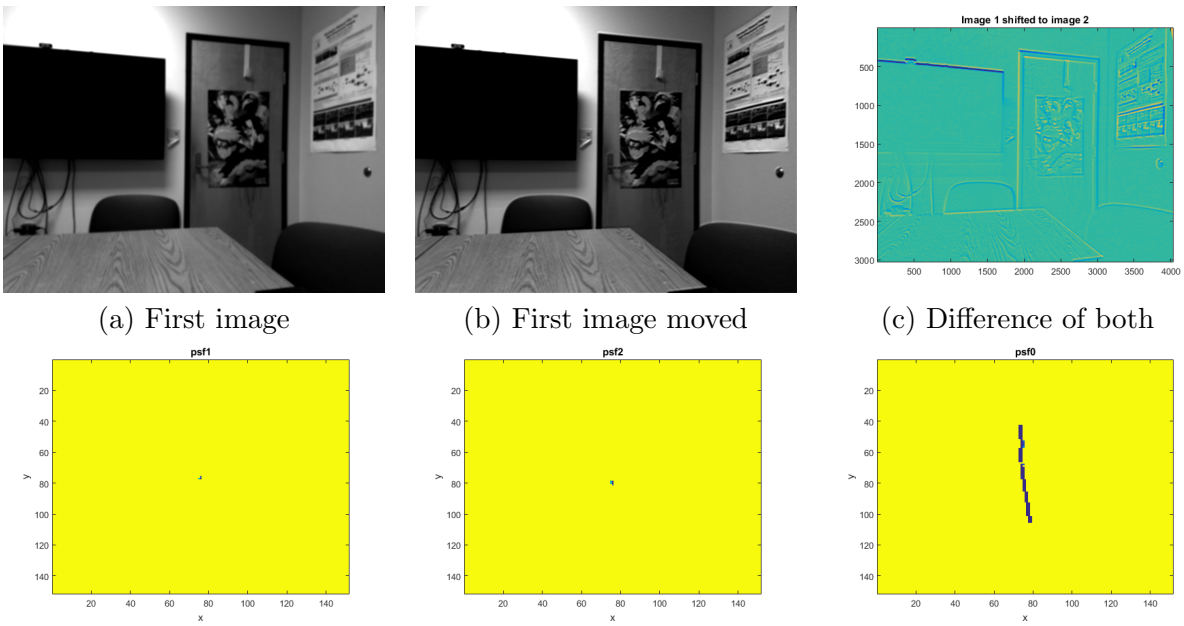
Figure 10.5: Example of recorded IMU θ_0 and image sensor $\{y_1, \ldots, y_N\}$ data in a burst shot mode of $N = 12$ images.

Having successfully deblurred the images using (8.7) gives hope to the proposed denoising method of (9.3). However, there are some differences we need to make a note of. First, each frame needs to have a short exposure time, and there should be no camera shake recorded during said time. This goes to say that although each frame is noisy, they are still sharp. Second, since each frame is sharp, the IMU data during the exposure time of each frame should be a simple dot (recording no movement), but over the whole burst, should depict the camera motion in between frames 1 and N. The point spread function of the reference image will be a centered dot, and that of the other frames will be a dot that has moved from the center.

10.2 Image Registration

To test the image registration, we first test the optical flow equation between two frames of a bust shot:

$$f(i,j,t+\Delta t) = f(i,j,t) - (\nabla f)^T \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} (H(t) - I) \begin{bmatrix} i \\ j \\ 1 \end{bmatrix}. \tag{10.1}$$



(a) First image (b) First image moved (c) Difference of both

(d) First frame (e) Second frame (f) Full burst

Figure 10.6: Example of image and their point spread functions

Figure 10.5 shows several noisy frames of a 12 image burst shot, along with the gyroscope recording over the full duration of the burst. Figure 10.6, on the other hand, shows the first image of the burst (a), along with the centered point spread function (d) which denotes its relative location (here it should be centered, because the reference is itself). (b) represents the shifted image of (a) using the homography matrix obtained from (e), and using equation (9.4). The difference between the reference image, and the shifted image based on the IMU homography is shows in (c). There is a clear misalignment, which means the gyroscope data is not accurate.

Due to the noise and drift the IMU generally suffers from, and as seen in Figure 10.6 (c), we have opted to use the distance transform to more accurately find the correct position of the camera for each frame of the burst shot.

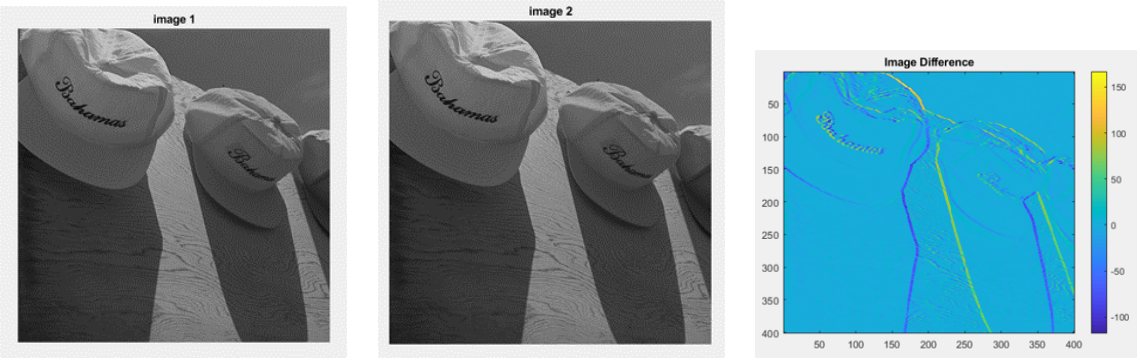
Knowing that the optical flow equation works as expected, the next step is to test whether the homography matrix derivation of Appendix B yields a good registration.

The first part, it to shift an image by a known amount. In this case, the shift was of 5 pixels in one direction with the following homography:

$$H_{true} = \begin{bmatrix} 1 & 0 & 5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10.2}$$

Figure 10.7 shows the two images, as well as the difference between both. The images being clean, registering them was successful as the homography obtained was:

$$\widehat{H} = \begin{bmatrix} 0.9829 & 7.637 \times 10^{-4} & 4.655 \\ -0.001 & 1.006 & 0.107 \\ 1.0515 \times 10^{-12} & -3.8053 \times 10^{-18} & 1 \end{bmatrix}. \tag{10.3}$$



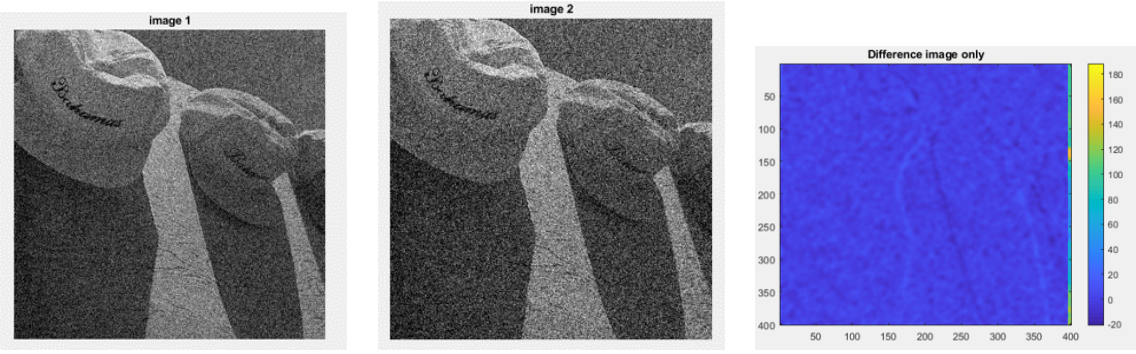
(a) Image (b) Shifted image (c) Difference

Figure 10.7: (a) reference image, (b) the image shifted by 5 pixels horizontally, and (c) the difference between both images.

Figure 10.8 shows the two images, as well as the difference between image 1 and image 2 registered to match image 1. In this case, synthetic noise was added to replicate a more realistic scenario of low light imaging. However, this registration relies solely on the image content, and with the added noise, is expected to fail. The resulting homography of the

registration attempt is:

$$\widehat{H} = \begin{bmatrix} 0.9796 & 0.0011 & 4.1962 \\ -0.0011 & 1.0014 & -0.245 \\ 1.2768 \times 10^{-13} & 5.8299 \times 10^{-16} & 1 \end{bmatrix}. \tag{10.4}$$



(a) Noisy image (b) Noisy shifted image (c) Registration difference

Figure 10.8: (a) noisy reference image, (b) the image shifted by 5 pixels horizontally, and (c) the difference between images after registration, without IMU homography.

The last test in Figure 10.9, is to give an H_{IMU} that is approximately equal to the true homography, but closer to what the gyroscope data would output. The result, as seen in the figure, is a perfect registration. The homography obtained is:

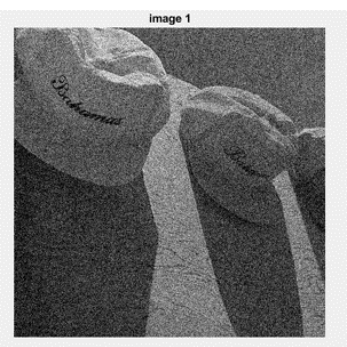
$$\widehat{H} = \begin{bmatrix} 0.9794 & 5.6842 \times 10^{-4} & 4.8559 \\ -0.0013 & 1.0002 & -0.0135 \\ -4.2814 \times 10^{-7} & -1.0939 \times 10^{-7} & 1.0004 \end{bmatrix}. \tag{10.5}$$

The image registration test being done, the only part left is image denoising.

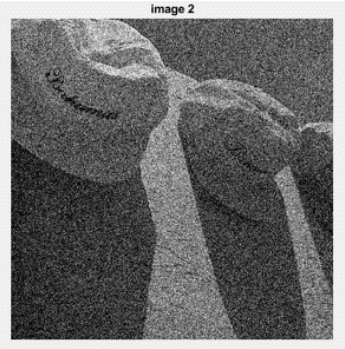
10.3 Image Denoising

Using the image denoising techniques proposed: Noise2Noise2MMSE and Noise2Noise2WSSIM, the results are seen in Figures 10.10 and 10.11.

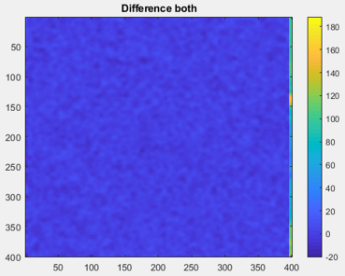
As expected, the method relying on structural similarity that uses 3 images, outputs a sharper result, whereas the method that uses the information in only two images yields a



(a) Noisy image

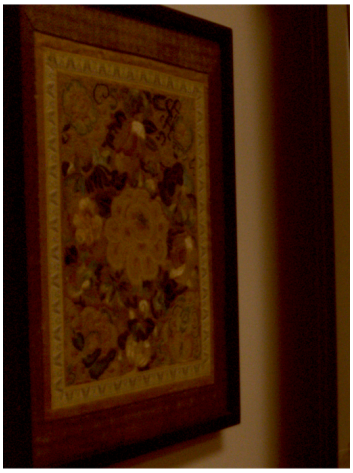


(b) Noisy shifted image



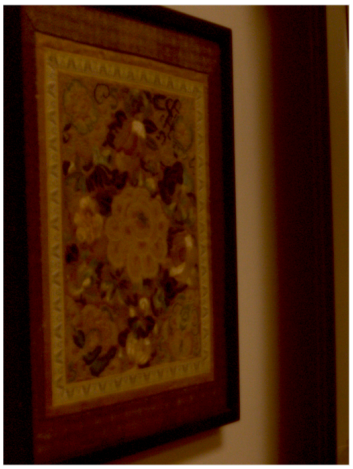
(c) Registration difference

Figure 10.9: (a) noisy reference image, (b) the image shifted by 5 pixels horizontally, and (c) the difference between images after registration, with IMU homography.



(a) Noisy image

(b) Noise2Noise2MMSE



(c) Noise2NoiseWSSIM

Figure 10.10: (a) noisy reference image of a frame, (b) denoised image using Noise2Noise2MMSE, (c) denoised image using Noise2Noise2WSSIM

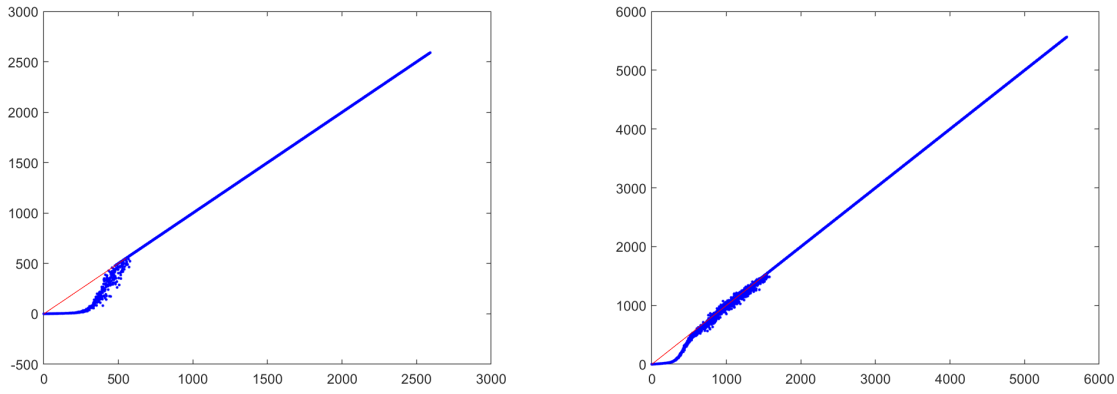
smoother image. This is reinforced if we look at the coring functions of both methods. The 45 degree line in the graph shows no change, and the larger the deviation from that line, the more smoothing is expected in the denoising function. Figure 10.12 shows the coring functions of both methods.

(a) Noisy image

(b) Noise2Noise2MMSE

(c) Noise2Noise2SSIM

Figure 10.11: (a) noisy reference image of a lightbox, (b) denoised image using Noise2Noise2MMSE, (c) denoised image using Noise2Noise2WSSIM



(a) Noise2Noise2MMSE

(b) Noise2NoiseWSSIM

Figure 10.12: Coring function of Noise2Noise2MMSE and Noise2Noise2SSIM

CHAPTER XI

CONCLUSIONS

11.1 BF and NLM

We proposed fast stochastic bilateral filter (FSBF) and fast stochastic non-local means (FSNLM), new methods aimed at reducing the complexity of the conventional bilateral filter and non-local means filter, respectively. FSBF and FSNLM combine the random filtering of multiple color channels into a single random convolutional filtering process, achieving the per-pixel complexity of $\mathcal{O}(L)$ where L is the number of random vectors drawn in Monte-Carlo. We proved theoretically and empirically that the Monte-Carlo convergence rate is invariant to the window size, the block size, and very slowly increasing with the increasing color dimension of the image. We further improved the convergence speed by introducing "quasi-random" numbers, implementing a faster Gaussian filter, and approximating NLM by BNLM.

11.2 Distance Transform

We proposed the notion of convolutional distance transform—a very close approximation to distance transform using convolution filters. Its complexity is $\mathcal{O}(N \log N)$ (or $\mathcal{O}(\tau N)$ or $\mathcal{O}(N)$ in special cases), which is faster than the $\mathcal{O}(N^2)$ brute-forced implementation and on par with the state-of-the-art implementations. But the convolutional distance transform is far simpler than the prior art—it requires only a few lines of Matlab code, agnostic to the type of distance metrics, and trivially generalizes to $L > 2$ dimensions. Our experiments verified the accuracy and speed of the proposed convolutional distance transforms.

11.3 Multi-frame Image Denoising

We proposed a novel method for image registration based on the optical flow method that leverages information from the noisy frames, as well as the IMU. It is more robust to noise than the traditional methods using feature extractions that rely only on image content. Furthermore, we proposed Noise2Noise2MMSE and Noise2Noise2WSSIM, two methods that use up to three registered frames for image denoising. The results obtained differ in sharpness levels based on the method used.

BIBLIOGRAPHY

[1] A. Olmos and F. A. Kingdom, "A biologically inspired algorithm for the recovery of shading and reflectance images," *Perception*, vol. 33, no. 12, pp. 1463–1473, 2004.

[2] K. N. Chaudhury, D. Sage, and M. Unser, "Fast O(1) bilateral filtering using trigonometric range kernels," *IEEE Trans. Image Process.*, vol. 20, no. 12, pp. 3376–82, Dec. 2011.

[3] K. Sugimoto, N. Fukushima, and S.-i. Kamata, "Fast bilateral filter for multichannel images via soft-assignment coding," in *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2016 Asia-Pacific.* IEEE, 2016, pp. 1–4.

[4] P. Nair and K. N. Chaudhury, "Fast high-dimensional filtering using clustering," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2017.

[5] A. Chakrabarti and T. Zickler, "Statistics of Real-World Hyperspectral Images," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 193–200.

[6] S. Paris and F. Durand, "A fast approximation of the bilateral filter using a signal processing approach," in *Computer Vision–ECCV 2006.* Springer, 2006, pp. 568–580.

[7] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Jan. 1998, pp. 839–846.

[8] V. Aurich and J. Weule, "Non-linear gaussian filters performing edge preserving diffusion," in *Mustererkennung 1995.* Springer, 1995, pp. 538–545.

[9] S. M. Smith and J. M. Brady, "Susana new approach to low level image processing," *International journal of computer vision*, vol. 23, no. 1, pp. 45–78, 1997.

[10] M. Zhang and B. K. Gunturk, "Multiresolution bilateral filtering for image denoising," *Image Processing, IEEE Transactions on*, vol. 17, no. 12, pp. 2324–2333, 2008.

[11] R. Keshet, D. Barash, D. Shaked, M. Elad, and R. Kimmel, "Bilateral filtering in a demosaicing process," Nov. 2004, uS Patent 6,816,197.

[12] S. Bae, S. Paris, and F. Durand, "Two-scale tone management for photographic look," in *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3. ACM, 2006, pp. 637–645.

[13] C. Richardt, D. Orr, I. Davies, A. Criminisi, and N. A. Dodgson, "Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid," in *European conference on Computer vision.* Springer, 2010, pp. 510–523.

[14] R. Crabb, C. Tracey, A. Puranik, and J. Davis, "Real-time foreground segmentation via range and color imaging," 2008.

[15] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2. IEEE, 2005, pp. 60–65.

[16] W. Cheng and K. Hirakawa, "Minimum risk wavelet shrinkage operator for poisson image denoising," *IEEE Transactions on Image Processing*, vol. 24, no. 5, pp. 1660–1671, 2015.

[17] C. Karam and K. Hirakawa, "Monte-Carlo acceleration of bilateral filter and non-local means," *IEEE Trans. Image Process.*, vol. 27, no. 3, pp. 1462–1474, Mar. 2018.

[18] K. Sugimoto and S. Kamata, "Compressive bilateral filtering," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 3357–3369, Nov. 2015.

[19] G. Deng, "Fast compressive bilateral filter," *Electronics Letters*, vol. 53, pp. 150–152, Feb. 2017.

[20] A. Dauwe, B. Goossens, H. Q. Luong, and W. Philips, "A fast non-local image denoising algorithm," in *Electronic Imaging 2008*. International Society for Optics and Photonics, 2008, pp. 681 210–681 210.

[21] S. H. Chan, T. Zickler, and Y. M. Lu, "Monte carlo non-local means: Random sampling for large-scale image filtering," 2014.

[22] B. Goossens, H. Luong, A. Pizurica, and W. Philips, "An improved non-local denoising algorithm," in *Local and Non-Local Approximation in Image Processing, International Workshop, Proceedings*, 2008, p. 143.

[23] C. Karam, K. Sugimoto, and K. Hirakawa, "Near-Constant time bilateral filter for high-dimensional images," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, 2018.

[24] L. Vincent and P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 6, pp. 583–598, 1991.

[25] O. Cuisenaire and B. Macq, "Fast and exact signed euclidean distance transformation with linear complexity," in *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, vol. 6. IEEE, 1999, pp. 3293–3296.

[26] Y. T. Chin, H. Wang, L. P. Tay, H. Wang, and W. Y. Soh, "Vision guided agv using distance transform," in *Proceedings of the 32nd ISR (International Symposium on Robotics)*, vol. 19. Citeseer, 2001, p. 21.

[27] F. Y. Shih and Y.-T. Wu, "Three-dimensional euclidean distance transformation and its application to shortest path planning," *Pattern Recognition*, vol. 37, no. 1, pp. 79–92, 2004.

[28] Y. Ge and J. M. Fitzpatrick, "On the generation of skeletons from discrete euclidean distance maps," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 11, pp. 1055–1066, 1996.

[29] P.-E. Danielsson, "Euclidean distance mapping," *Computer Graphics and image processing*, vol. 14, no. 3, pp. 227–248, 1980.

[30] M. Couprie, A. Saude, and G. Bertrand, "Euclidean homotopic skeleton based on critical kernels," in *2006 19th Brazilian Symposium on Computer Graphics and Image Processing*. IEEE, 2006, pp. 307–314.

[31] D. Coeurjolly and A. Montanvert, "Optimal separable algorithms to compute the reverse euclidean distance transformation and discrete medial axis in arbitrary dimension," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 3, 2007.

[32] G. Borgefors, "Distance transformations in digital images," *Computer vision, graphics, and image processing*, vol. 34, no. 3, pp. 344–371, 1986.

[33] D. W. Paglieroni, "Distance transforms: Properties and machine vision applications," *CVGIP: Graphical models and image processing*, vol. 54, no. 1, pp. 56–74, 1992.

[34] H.-C. Liu and M. D. Srinath, "Partial shape classification using contour matching in distance transformation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 11, pp. 1072–1079, 1990.

[35] J. You, E. Pissaloux, W. Zhu, and H. A. Cohen, "Efficient image matching: A hierarchical chamfer matching scheme via distributed system," *Real-Time Imaging*, vol. 1, no. 4, pp. 245–259, 1995.

[36] T. B. Sebastian, H. Tek, J. J. Crisco, and B. B. Kimia, "Segmentation of carpal bones from ct images using skeletally coupled deformable models," *Medical Image Analysis*, vol. 7, no. 1, pp. 21–45, 2003.

[37] C. Jia and B. L. Evans, "Online calibration and synchronization of cellphone camera and gyroscope," in *2013 IEEE Global Conference on Signal and Information Processing*. IEEE, 2013, pp. 731–734.

[38] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising with block-matching and 3d filtering," in *Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning*, vol. 6064. International Society for Optics and Photonics, 2006, p. 606414.

[39] A. Buades, B. Coll, and J.-M. Morel, "Non-local means denoising," *Image Processing On Line*, vol. 1, pp. 208–212, 2011.

[40] S. Guo, Z. Yan, K. Zhang, W. Zuo, and L. Zhang, "Toward convolutional blind denoising of real photographs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1712–1722.

[41] A. Krull, T.-O. Buchholz, and F. Jug, "Noise2void-learning denoising from single noisy images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2129–2137.

[42] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, and T. Aila, "Noise2noise: Learning image restoration without clean data," *arXiv preprint arXiv:1803.04189*, 2018.

[43] K. Sugimoto and S. Kamata, "Efficient constant-time Gaussian filtering with sliding DCT/DST-5 and dual-domain error minimization," *ITE Trans. Media Technol. Appl.*, vol. 3, no. 1, pp. 12–21, 2015.

[44] K. Sugimoto, S. Kyochi, and S. Kamata, "Universal approach for DCT-based constant-time Gaussian filter with moment preservation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Apr. 2018.

[45] R. Deriche, "Fast algorithms for low-level vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 1, pp. 78–87, 1990.

[46] I. T. Young and L. J. van Vliet, "Recursive implementation of the Gaussian filter," *Signal Process.*, vol. 44, no. 2, pp. 139–151, Jun. 1995.

[47] L. J. van Vliet, I. T. Young, and P. W. Verbeek, "Recursive Gaussian derivative filters," in *Proc. Int. Conf. Pattern Recognition (ICPR)*, vol. 1, no. 1, 1998, pp. 509–514.

[48] G. Farnebäck and C. Westin, "Improving Deriche-style recursive Gaussian filters," *J. Math. Imaging and Vis.*, vol. 26, no. 3, pp. 293–299, Nov. 2006.

[49] K. Sugimoto and S. Kamata, "Fast Gaussian filter with second-order shift property of DCT-5," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2013, pp. 514–518.

[50] D. Charalampidis, "Recursive implementation of the Gaussian filter using truncated cosine functions," *IEEE Trans. Signal Process.*, vol. 64, no. 14, pp. 3554–3565, 2016.

[51] G. Levy, "An introduction to quasi-random numbers," *Numerical Algorithms Group Ltd.*, p. 143, 2002.

[52] U. Montanari, "A method for obtaining skeletons using a quasi-euclidean distance," *Journal of the ACM (JACM)*, vol. 15, no. 4, pp. 600–624, 1968.

[53] B. J. H. Verwer, P. W. Verbeek, and S. T. Dekker, "An efficient uniform cost algorithm applied to distance transforms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 4, pp. 425–429, 1989.

[54] I. Ragnemalm, "Neighborhoods for distance transformations using ordered propagation," *CVGIP: Image Understanding*, vol. 56, no. 3, pp. 399–409, 1992.

[55] H. Eggers, "Two fast euclidean distance transformations in z2based on sufficient propagation," *Computer Vision and Image Understanding*, vol. 69, no. 1, pp. 106–116, 1998.

[56] O. Cuisenaire and B. Macq, "Fast euclidean distance transformations by propagation using multiple neighbourhoods," *Computer Vision and Image Understanding*, vol. 76, no. EPFL-ARTICLE-86619, p. 163, 1999.

[57] A. X. Falcão, J. Stolfi, and R. de Alencar Lotufo, "The image foresting transform: Theory, algorithms, and applications," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 1, pp. 19–29, 2004.

[58] G. Borgefors, "Distance transformations in arbitrary dimensions," *Computer vision, graphics, and image processing*, vol. 27, no. 3, pp. 321–345, 1984.

[59] M. N. Kolountzakis and K. N. Kutulakos, "Fast computation of the euclidian distance maps for binary images," *Information processing letters*, vol. 43, no. 4, pp. 181–184, 1992.

[60] L. Chen, "A fast algorithm for euclidian distance maps of a 2-d binary image," *Inf. Process. Lett.*, vol. 51, no. 1, pp. 25–29, 1994.

[61] T. Hirata, "A unified linear-time algorithm for computing distance maps," *Inf. Process. Lett.*, vol. 58, no. 3, pp. 129–133, 1996.

[62] T. Saito and J.-I. Toriwaki, "New algorithms for euclidean distance transformation of an n-dimensional digitized picture with applications," *Pattern recognition*, vol. 27, no. 11, pp. 1551–1565, 1994.

[63] F.-C. Shih and O. R. Mitchell, "A mathematical morphology approach to euclidean distance transformation," *IEEE Transactions on Image Processing*, vol. 1, no. 2, pp. 197–204, 1992.

[64] C. T. Huang and O. R. Mitchell, "A euclidean distance transform using grayscale morphology decomposition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 4, pp. 443–448, 1994.

[65] R. A. Lotufo and F. A. Zampirolli, "Fast multidimensional parallel euclidean distance transform based on mathematical morphology," in *Computer Graphics and Image Processing, 2001 Proceedings of XIV Brazilian Symposium on.* IEEE, 2001, pp. 100–105.

[66] S. Fortune, "A sweepline algorithm for voronoi diagrams," *Algorithmica*, vol. 2, no. 1-4, p. 153, 1987.

[67] H. Breu, J. Gil, D. Kirkpatrick, and M. Werman, "Linear time euclidean distance transform algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 5, pp. 529–533, 1995.

[68] R. L. Ogniewicz and O. Kübler, "Voronoi tessellation of points with integer coordinates: Time-efficient implementation and online edge-list generation," *Pattern Recognition*, vol. 28, no. 12, pp. 1839–1844, 1995.

[69] C. R. Maurer, R. Qi, and V. Raghavan, "A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 265–270, 2003.

[70] K. Sugimoto, S. Kyochi, and S.-I. Kamata, "Universal approach for dct-based constant-time gaussian filter with moment preservation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 1498–1502.

[71] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for markov random fields with smoothness-based priors," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 6, pp. 1068–1080, 2008.

[72] L. Zhang and X. Wu, "An edge-guided image interpolation algorithm via directional filtering and data fusion," *IEEE transactions on Image Processing*, vol. 15, no. 8, pp. 2226–2238, 2006.

[73] Z. Farbman, R. Fattal, D. Lischinski, and R. Szeliski, "Edge-preserving decompositions for multi-scale tone and detail manipulation," in *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3. ACM, 2008, p. 67.

[74] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactions on Image processing*, vol. 15, no. 12, pp. 3736–3745, 2006.

[75] M. K. Mihcak, I. Kozintsev, K. Ramchandran, and P. Moulin, "Low-complexity image denoising based on statistical modeling of wavelet coefficients," *IEEE Signal Processing Letters*, vol. 6, no. 12, pp. 300–303, 1999.

[76] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE transactions on Computers*, vol. 100, no. 1, pp. 90–93, 1974.

[77] L. Zhang, W. Dong, D. Zhang, and G. Shi, "Two-stage image denoising by principal component analysis with local pixel grouping," *Pattern recognition*, vol. 43, no. 4, pp. 1531–1549, 2010.

[78] C. Liu and W. T. Freeman, "A high-quality video denoising algorithm based on reliable motion estimation," in *European conference on computer vision*. Springer, 2010, pp. 706–719.

[79] A. Karpenko, D. Jacobs, J. Baek, and M. Levoy, "Digital video stabilization and rolling shutter correction using gyroscopes," *CSTR*, vol. 1, p. 2, 2011.

[80] K. J. Barnard, C. E. White, and A. E. Absi, "Two-dimensional restoration of motion-degraded intensified ccd imagery," *Applied optics*, vol. 38, no. 10, pp. 1942–1952, 1999.

[81] N. Joshi, S. B. Kang, C. L. Zitnick, and R. Szeliski, "Image deblurring using inertial measurement sensors," in *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4. ACM, 2010, p. 30.

[82] H. Bae, C. C. Fowlkes, and P. H. Chou, "Accurate motion deblurring using camera motion tracking and scene depth," in *2013 IEEE Workshop on Applications of Computer Vision (WACV)*. IEEE, 2013, pp. 148–153.

[83] O. Sindelar and F. Sroubek, "Image deblurring in smartphone devices using built-in inertial measurement sensors," *Journal of Electronic Imaging*, vol. 22, no. 1, p. 011003, 2013.

[84] S. Hee Park and M. Levoy, "Gyro-based multi-image deconvolution for removing handshake blur," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3366–3373.

[85] T. Chen, K. Yamamoto, S. Chhatkuli, and H. Shimamura, "Panoramic epipolar image generation for mobile mapping system," *Proceedings of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 39, p. B5, 2012.

[86] S. Zhang and R. L. Stevenson, "Inertia sensor aided alignment for burst pipeline in low light conditions," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 3953–3957.

[87] X.-J. Mao, C. Shen, and Y.-B. Yang, "Image restoration using convolutional auto-encoders with symmetric skip connections," *arXiv preprint arXiv:1606.08921*, 2016.

[88] Y. Zhang and K. Hirakawa, "Combining inertial measurements with blind image deblurring using distance transform," *IEEE Transactions on Computational Imaging*, vol. 2, no. 3, pp. 281–293, 2016.

[89] M. Protter and M. Elad, "Image sequence denoising via sparse and redundant representations," *IEEE Transactions on Image Processing*, vol. 18, no. 1, pp. 27–35, 2008.

[90] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

APPENDIX A

Proof of Corollary 1

Proof. Regarding the variance of $\cos(\boldsymbol{\zeta}^T\boldsymbol{Z})$, the proof is found in [17]. Let $\boldsymbol{q} \in \mathbb{R}^C$ be a vector. By basic trigonometry, we have

$$\begin{aligned}\mathbb{E}\left[\left(-\boldsymbol{\zeta}\sin\left(\boldsymbol{\zeta}^T\boldsymbol{q}\right)\right)\left(-\boldsymbol{\zeta}\sin\left(\boldsymbol{\zeta}^T\boldsymbol{q}\right)\right)^T\right] &= \mathbb{E}\left[\frac{1-\cos\left(2\boldsymbol{\zeta}^T\boldsymbol{q}\right)}{2}\cdot\boldsymbol{\zeta}\boldsymbol{\zeta}^T\right] \\ &= \frac{\Theta^{-1}}{2} + \frac{1}{8}\nabla^2\mathbb{E}\left[\cos\left(2\boldsymbol{\zeta}^T\boldsymbol{q}\right)\right],\end{aligned} \tag{A.1}$$

where ∇^2 is the Laplace operator defined over the vector $\boldsymbol{q} \in \mathbb{R}^C$, and the last equality stems from the relation

$$\nabla^2\cos\left(2\boldsymbol{\zeta}^T\boldsymbol{q}\right) = -\nabla 2\boldsymbol{\zeta}\sin\left(2\boldsymbol{\zeta}^T\boldsymbol{q}\right) = -4\boldsymbol{\zeta}\boldsymbol{\zeta}^T\cos\left(2\boldsymbol{\zeta}^T\boldsymbol{q}\right). \tag{A.2}$$

Invoking Lemma 2 yields the following:

$$\begin{aligned}&\mathbb{E}\left[\left(-\boldsymbol{\zeta}\sin\left(\boldsymbol{\zeta}^T\boldsymbol{q}\right)\right)\left(-\boldsymbol{\zeta}\sin\left(\boldsymbol{\zeta}^T\boldsymbol{q}\right)\right)^T\right] \\ &= \frac{\Theta^{-1}}{2} + \frac{1}{8}\nabla^2\exp\left(-\frac{4\boldsymbol{q}^T\Theta^{-1}\boldsymbol{q}}{2}\right) \\ &= \Theta^{-1}\frac{1-\exp\left(-2\boldsymbol{q}^T\Theta^{-1}\boldsymbol{q}\right)}{2} + 2\Theta^{-1}\boldsymbol{q}\boldsymbol{q}^T\Theta^{-1}\exp\left(-2\boldsymbol{q}^T\Theta^{-1}\boldsymbol{q}\right).\end{aligned} \tag{A.3}$$

Similarly, we obtain from (A.2) and Lemma 2 the relation

$$\begin{aligned}\mathbb{E}\left[-\boldsymbol{\zeta}\sin\left(\boldsymbol{\zeta}^T\boldsymbol{q}\right)\right] &= \nabla\mathbb{E}\left[\cos(\zeta^T\boldsymbol{q})\right] \\ &= -\Theta^{-1}\boldsymbol{q}\exp\left(-\frac{\boldsymbol{q}^T\Theta\boldsymbol{q}}{2}\right).\end{aligned} \tag{A.4}$$

Substituting (3.23), we obtain the covariance matrix:

$$\begin{aligned}&\mathbb{E}\left[\left(-\boldsymbol{\zeta}\sin\left(\boldsymbol{\zeta}^T\boldsymbol{q}\right)\right)\left(-\boldsymbol{\zeta}\sin\left(\boldsymbol{\zeta}^T\boldsymbol{q}\right)\right)^T\right] \\ &- \left(\mathbb{E}\left[-\boldsymbol{\zeta}\sin\left(\boldsymbol{\zeta}^T\boldsymbol{q}\right)\right]\right)\left(\mathbb{E}\left[-\boldsymbol{\zeta}\sin\left(\boldsymbol{\zeta}^T\boldsymbol{q}\right)\right]\right)^T \\ &= \Theta^{-1}\frac{1-\alpha^2}{2} + \Theta^{-1}\boldsymbol{q}\boldsymbol{q}^T\Theta^{-1}(2\alpha^2-\alpha).\end{aligned} \tag{A.5}$$

□

APPENDIX B

Solution of Homography Matrix

Proposed denoising method:

$$\{\widehat{f},\widehat{H}\} = \arg\min_{f,H} \sum_{t=1}^{N} \left\| g(i,j,t) - f\left(H(t) \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} \right) \right\|^2 + \lambda \|H(t) - H_{IMU}(t)\|_F^2 + \alpha E_1(f) \tag{B.1}$$

Optical flow equation:

$$f(i+\Delta i, j+\Delta j, t+\Delta t) = f(i,j,t) + \Delta i f_i(i,j,t) + \Delta j f_j(i,j,t) + \Delta t f(i,j,t) \tag{B.2}$$

$$f(i,j,t+\Delta t) = f\left(H(t) \begin{bmatrix} i \\ j \\ 1 \end{bmatrix}, t \right) \tag{B.3}$$

$$\begin{bmatrix} i+\Delta i \\ j+\Delta j \\ 1 \end{bmatrix} = \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} + \begin{bmatrix} \Delta i \\ \Delta j \\ 0 \end{bmatrix} = H(t) \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} \tag{B.4}$$

$$\begin{bmatrix} \Delta i \\ \Delta j \\ 0 \end{bmatrix} = H(t) \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} - I \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} = (H(t) - I) \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} \tag{B.5}$$

$$f(i,j,t+\Delta t) = f\left(H(t) \begin{bmatrix} i \\ j \\ 1 \end{bmatrix}, t \right) \tag{B.6}$$

$$f(i,j,t+\Delta t) = f(i,j,t) - (\nabla f)^T \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} (H(t) - I) \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} \tag{B.7}$$

Replacing (8) in (1) allows us to split the problem in two.

Getting the homography cost: (one number)

$$\begin{aligned} J = & \Bigg[g - f - diag(i) \left(\nabla f\right)^T M_{ij} \left(H(t) - I\right) M_i - diag(j) \left(\nabla f\right)^T M_{ij} \left(H(t) - I\right) M_j \\ & - \left(\nabla f\right)^T M_{ij} \left(H(t) - I\right) M_k \Bigg]^T \Bigg[g - f - diag(i) \left(\nabla f\right)^T M_{ij} \left(H(t) - I\right) M_i \\ & - diag(j) \left(\nabla f\right)^T M_{ij} \left(H(t) - I\right) M_j - \left(\nabla f\right)^T M_{ij} \left(H(t) - I\right) M_k \Bigg] \\ & + \lambda \| H(t) - H_{IMU}(t) \|_F^2, \end{aligned} \tag{B.8}$$

where $M_{ij} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$, $M_i = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$, $M_j = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$, and $M_k = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$.

$$\begin{aligned} J = & g^T g + f^T f + M_i^T \left(H(t) - I\right)^T M_{ij}^T \left(\nabla f\right) diag(i) diag(i) \left(\nabla f\right)^T M_{ij} \left(H(t) - I\right) M_i \\ & + M_j^T \left(H(t) - I\right)^T M_{ij}^T \left(\nabla f\right) diag(j) diag(j) \left(\nabla f\right)^T M_{ij} \left(H(t) - I\right) M_j \\ & + M_k^T \left(H(t) - I\right)^T M_{ij}^T \left(\nabla f\right) \left(\nabla f\right)^T M_{ij} \left(H(t) - I\right) M_k \\ & + 2 \Bigg[-g^T f - g^T diag(i) \left(\nabla f\right)^T M_{ij} \left(H(t) - I\right) M_i - g^T diag(j) \left(\nabla f\right)^T M_{ij} \left(H(t) - I\right) M_j \\ & - g^T \left(\nabla f\right)^T M_{ij} \left(H(t) - I\right) M_k + f^T diag(i) \left(\nabla f\right)^T M_{ij} \left(H(t) - I\right) M_i \\ & + f^T diag(j) \left(\nabla f\right)^T M_{ij} \left(H(t) - I\right) M_j + f^T \left(\nabla f\right)^T M_{ij} \left(H(t) - I\right) M_k \\ & + M_i^T \left(H(t) - I\right)^T M_{ij}^T \left(\nabla f\right) diag(i) diag(j) \left(\nabla f\right)^T M_{ij} \left(H(t) - I\right) M_j \\ & + M_i^T \left(H(t) - I\right)^T M_{ij}^T \left(\nabla f\right) diag(i) \left(\nabla f\right)^T M_{ij} \left(H(t) - I\right) M_k \\ & + M_j^T \left(H(t) - I\right)^T M_{ij}^T \left(\nabla f\right) diag(j) \left(\nabla f\right)^T M_{ij} \left(H(t) - I\right) M_k \Bigg] \\ & + \lambda \| \left(H(t) - I\right) - \left(H_{IMU} - I\right) \|_F^2 \end{aligned} \tag{B.9}$$

Taking the derivative with respect to $(H(t) - I)$ of the cost function above, we get:

$$
\begin{aligned}
\frac{\partial J}{\partial \left(H(t)-I\right)} &= 2\left\{M_{ij}^{T}\left(\nabla f\right) diag(i) diag(i)\left(\nabla f\right) M_{ij}\right\}\left(H(t)-I\right) M_i M_i^{T} \\
&+ 2\left\{M_{ij}^{T}\left(\nabla f\right) diag(j) diag(j)\left(\nabla f\right) M_{ij}\right\}\left(H(t)-I\right) M_j M_j^{T} \\
&+ 2\left\{M_{ij}^{T}\left(\nabla f\right)\left(\nabla f\right) M_{ij}\right\}\left(H(t)-I\right) M_k M_k^{T} \\
&- 2\left[g^{T} diag(i)\left(\nabla f\right)^{T} M_{ij}\right]^{T} M_i^{T} - 2\left[g^{T} diag(j)\left(\nabla f\right)^{T} M_{ij}\right]^{T} M_j^{T} \\
&- 2\left[g^{T}\left(\nabla f\right)^{T} M_{ij}\right]^{T} M_k^{T} + 2\left[f^{T} diag(i)\left(\nabla f\right)^{T} M_{ij}\right]^{T} M_i^{T} \\
&+ 2\left[f^{T} diag(j)\left(\nabla f\right)^{T} M_{ij}\right]^{T} M_j^{T} + 2\left[f^{T}\left(\nabla f\right)^{T} M_{ij}\right]^{T} M_k^{T} \\
&+ \left[M_{ij}^{T}\left(\nabla f\right) diag(i) diag(j)\left(\nabla f\right)^{T} M_{ij}\right]^{T}\left(H(t)-I\right) M_i M_j^{T} \\
&+ \left[M_{ij}^{T}\left(\nabla f\right) diag(i) diag(j)\left(\nabla f\right)^{T} M_{ij}\right]^{T}\left(H(t)-I\right) M_j M_i^{T} \\
&+ \left[M_{ij}^{T}\left(\nabla f\right) diag(i)\left(\nabla f\right)^{T} M_{ij}\right]^{T}\left(H(t)-I\right) M_j M_k^{T} \\
&+ \left[M_{ij}^{T}\left(\nabla f\right) diag(i)\left(\nabla f\right)^{T} M_{ij}\right]^{T}\left(H(t)-I\right) M_k M_i^{T} \\
&+ \left[M_{ij}^{T}\left(\nabla f\right) diag(j)\left(\nabla f\right)^{T} M_{ij}\right]^{T}\left(H(t)-I\right) M_j M_k^{T} \\
&+ \left[M_{ij}^{T}\left(\nabla f\right) diag(j)\left(\nabla f\right)^{T} M_{ij}\right]^{T}\left(H(t)-I\right) M_k M_j^{T} \\
&+ 2\lambda\left(\left(H(t)-I\right)-\left(H_{IMU}-I\right)\right)
\end{aligned}
\tag{B.10}
$$

Setting the previous equation to zero allows us to minimize the error and solve to obtain the optical flow homography matrix $H(t)$. To solve it, we use the following equation: $\sum_n A_n X B_n = c$ whose solution is: $vec(X) = \left(\sum_n B_n^T \otimes A_n\right)^{-1} vec(c)$, where $\otimes$ represents the Kronecker product.

$$
\begin{aligned}
\sum_n B_n^T \otimes A_n = & \left[M_i M_i^T\right] \otimes 2\left[M_{ij}^T \left(\nabla f\right) diag(i) diag(i) \left(\nabla f\right)^T M_{ij}\right] \\
& + \left[M_j M_j^T\right] \otimes 2\left[M_{ij}^T \left(\nabla f\right) diag(j) diag(j) \left(\nabla f\right)^T M_{ij}\right] \\
& + \left[M_k M_k^T\right] \otimes 2\left[M_{ij}^T \left(\nabla f\right) \left(\nabla f\right)^T M_{ij}\right] \\
& + \left[M_i M_j^T\right] \otimes \left[M_{ij}^T \left(\nabla f\right) diag(i) diag(j) \left(\nabla f\right)^T M_{ij}\right] \\
& + \left[M_j M_i^T\right] \otimes \left[M_{ij}^T \left(\nabla f\right) diag(i) diag(j) \left(\nabla f\right)^T M_{ij}\right] \\
& + \left[M_i M_k^T\right] \otimes \left[M_{ij}^T \left(\nabla f\right) diag(i) \left(\nabla f\right)^T M_{ij}\right] \\
& + \left[M_k M_i^T\right] \otimes \left[M_{ij}^T \left(\nabla f\right) diag(i) \left(\nabla f\right)^T M_{ij}\right] \\
& + \left[M_j M_k^T\right] \otimes \left[M_{ij}^T \left(\nabla f\right) diag(j) \left(\nabla f\right)^T M_{ij}\right] \\
& + \left[M_k M_j^T\right] \otimes \left[M_{ij}^T \left(\nabla f\right) diag(j) \left(\nabla f\right)^T M_{ij}\right] \\
& + I \otimes 2\lambda
\end{aligned}
\tag{B.11}
$$

$$
\begin{aligned}
c = & 2\left[g^T diag(i) \left(\nabla f\right)^T M_{ij}\right]^T M_i^T + 2\left[g^T diag(j) \left(\nabla f\right)^T M_{ij}\right]^T M_j^T \\
& + 2\left[g^T \left(\nabla f\right)^T M_{ij}\right]^T M_k^T - 2\left[f^T diag(i) \left(\nabla f\right)^T M_{ij}\right]^T M_i^T \\
& - 2\left[f^T diag(j) \left(\nabla f\right)^T M_{ij}\right]^T M_j^T - 2\left[f^T \left(\nabla f\right)^T M_{ij}\right]^T M_k^T \\
& + 2\lambda(H_{IMU} - I)
\end{aligned}
\tag{B.12}
$$