High Performance 4D Light Field Disparity Estimation, Super-Resolution and Compression

Von der Fakultät für Informatik, Elektrotechnik und Informationstechnik

der Universität Stuttgart zur Erlangung der Würde eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigte Abhandlung

Vorgelegt von

Trung Hieu Tran

aus Hanoi, Vietnam

Hauptberichter:	Professor Dr.-Ing. Sven Simon
Mitberichter:	Professor Dr. Hassan Rabah
Tag der mündlichen Prüfung:	14.12.2022

Institut für Technische Informatik

der Universität Stuttgart

2022

To my beloved family

Abstract

This dissertation investigates the high-performance processing of 4D light field (LF) as a combination of fast computation and high-quality output. It presents various contributions revolving around the three essential processing tasks of 4DLF: disparity estimation, super-resolution, and compression.

For the disparity estimation task, the goal is to develop a computational approach which is not only fast in processing time but also able to obtain accurate disparity information. For this purpose, we applied in the first step the variational computation framework from optical flow literature to estimate a 4DLF disparity map which is then enhanced in a post-processing step. The proposed approach greatly benefits from the intrinsic pixel precision of variational formulation and the effectiveness of weighted median filtering as a post-processing technique to achieve a highly accurate solution. Solving the variational optimization problem requires a nested iterative computation over multi-scale 4DLF causing a huge burden for CPU computation. The fast processing time is achieved by taking advantage of parallel computation on Graphic Processing Units (GPUs) to alleviate this high computational demand and at the same time efficiently handle the weighted median filtering operator.

For the super-resolution task, two approaches are developed in this work. The first one is an optimization-based approach dealing with the spatial super-resolution of 4DLF under mixed noise conditions. The second one is a deep learning-based approach focusing on a more general scenario, in which the reconstruction of high-resolution LF is carried out on spatial, angular and spatial-angular dimensions. In the first approach, a super-resolution model derived from a statistical perspective is presented. It assembles a joint $\ell^1 - \ell^2$ data fidelity term and a weighted regularization term. The non-smooth convex optimization problem is effectively solved by the alternating direction method of multipliers algorithm. It is also shown that GPU acceleration is well-suited to speed up the iterative solving process and results in a significant speed-up as compared to CPU execution. In the second approach, 4DLF is sliced into 3D epipolar (EPI) volumes, and a two-stage deep learning framework is applied to reconstruct high-resolution volumes. Through 3D convolutional operations and efficient deep-learning architecture, the proposed network makes use of angular and spatial information presented in the 3D EPI volume structure to reconstruct high-frequency details. As a result, the reconstructed high-resolution LF demonstrates a balanced performance distribution across all perspectives and presents superior visual quality compared to the previous works.

For the 4DLF compression task, our work focus on exploiting the content similarity existing in the perspective images of 4DLF to improve encoding performance. A motion-compensated wavelet decomposition scheme is therefore proposed to decompose 4DLF into high-/low-pass sub-band views in which the redundancy of image data was eliminated. An application of a standard coding tool, i.e. JPEG2000, demonstrates a clear improvement in compression performance in both lossless and lossy compression scenarios. In addition, GPU acceleration applied to the time-consuming lifting procedure shows a significant gain in processing time.

Zusammenfassung

In dieser Dissertation wird die hochperformante Verarbeitung von 4D-Lichtfeldern (LF) als Kombination aus schneller Berechnung und hochwertiger Ausgabe untersucht. Es werden verschiedene Beiträge vorgestellt, die sich um die drei wesentlichen Verarbeitungsaufgaben von 4DLF drehen: Disparitätsschätzung, Super-Resolution und Kompression.

Für die Disparitätsschätzung wurde ein Berechnungsansatz entwickelt, der nicht nur hinsichtlich der Verarbeitungszeit schnell ist, sondern auch in der Lage ist, eine genaue Disparitätsinformationen zu ermitteln. Zu diesem Zweck haben wir in einem ersten Schritt die Variationsberechnung aus der Literatur zum optischen Fluss angewendet, um eine 4DLF-Disparitätskarte zu schätzen, die dann in einem Nachverarbeitungsschritt verbessert wird. Der vorgeschlagene Ansatz profitiert in hohem Maße von der intrinsischen Pixelgenauigkeit der Variationsformulierung und der Wirksamkeit der gewichteten Medianfilterung als Nachbearbeitungstechnik, um eine hochgenaue Lösung zu erzielen. Die Lösung des Variationsoptimierungsproblems erfordert eine iterative Berechnung der 4D-Lichtfeldern auf mehreren Skalen, was eine enorme Belastung für die CPU darstellt. Die schnelle Verarbeitungszeit wird erreicht, indem die Vorteile der parallelen Berechnung auf Grafikprozessoren (GPUs) genutzt werden, um die Rechenzeit zu verringern und gleichzeitig den gewichteten Medianfilteroperator effizient zu verarbeiten.

Für die Fragestellung der Super-Resolution werden in dieser Arbeit zwei Ansätze entwickelt. Der erste ist ein optimierungsbasierter Ansatz, der sich mit der räumlichen Super-Resolution von 4D-Lichtfeldern bei Rauschen mit verschiedenen Charakteristika befasst. Der zweite ist ein auf Deep Learning basierender Ansatz, der sich auf ein allgemeineres Szenario konzentriert, in dem die Rekonstruktion hochauflösender Lichtfelder auf vier Dimensionen durchgeführt wird. Im ersten Ansatz wird ein Super-Resolution-Modell vorgestellt, das aus einer statistischen Perspektive abgeleitet ist. Es setzt sich aus einem gemeinsamen $\ell^1 - \ell^2$-Term zur Datentreue und einem gewichteten Regularisierungsterm zusammen. Das nicht-glatte konvexe Optimierungsproblem wird effektiv durch den Algorithmus der alternierenden Richtungsmethode der Multiplikatoren gelöst. Es wird auch gezeigt, dass die GPU-Beschleunigung gut geeignet ist, um den iterativen Lösungsprozess zu beschleunigen und zu einer signifikanten Beschleunigung im Vergleich zur CPU-Ausführung führt. Im zweiten Ansatz wird 4DLF in epipolare 3D-Volumina (EPI) zerlegt und ein zweistufiges Deep-Learning-Verfahren angewendet, um hochauflösende Volumina zu rekonstruieren. Durch 3D-Faltungs-operationen und eine effiziente Deep-Learning-Architektur nutzt das vorgeschlagene Netzwerk die in der 3D-EPI-Volumenstruktur enthaltenen Direktional- und Rauminformationen, um hochfrequente Details zu rekonstruieren. Als Ergebnis zeigt die rekonstruierte hochauflösende LF eine ausgewogene Leistungsverteilung über alle Perspektiven hinweg und präsentiert eine überlegene visuelle Qualität im Vergleich zu den bisherigen Arbeiten.

Für die 4DLF-Kompression konzentriert sich unsere Arbeit auf die Ausnutzung der inhaltlichen Ähnlichkeit, die in den perspektivischen Bildern der 4DLF besteht, um die Kompressionsleistung zu verbessern. Daher wird ein bewegungskompensiertes Wavelet-Zerlegungsschema vorgeschlagen, um 4DLF in Hoch-/Tiefpass-Subbandansichten zu zerlegen, bei denen die Redundanz der Bilddaten eliminiert wurde. Die Anwendung eines Standard-Codierungs-Methode wie JPEG2000, zeigt eine deutliche Verbesserung der Kompressionsleistung sowohl in verlustfreien als auch in verlustbehafteten Kompressionsszenarien. Darüber hinaus zeigt die GPU-Beschleunigung des zeitaufwändigen Lifting-Verfahrens einen deutlichen Gewinn bezüglich der Verarbeitungszeit.

Acknowledgements

This dissertation would not have reached its conclusion if it were not for the support of the kind people around me, to only some of whom it is possible to give a particular mention here.

Firstly, I would like to express my sincere thanks to my advisor and mentor, Prof. Dr-Ing. Sven Simon, for supervising me during this thesis and even more for entrusting me with this research position. Without his patience in listening to me during our first call and his hard-working over the weekend, I may not make my way to be in Germany and work on this dissertation.

I would like to acknowledge the financial support from the German Academic Exchange Service (DAAD) and Baden-Württemberg Stiftung without which I would not be able to work on my dissertation.

My PhD life would be rough and tedious without the warm, joyful but fruitful working environment for which I would like to thank my colleagues at the Parallel System Department and later at the Computational Imaging System department. Thanks to Jajnabalkya Guhathakurta, Yousef Baroud, Gasim Mammadov and Timo Schweizer for sharing with me their interest in table tennis. I truly missed the competitive atmosphere we did have together after working time. Many thanks to Steffen Kieß for his as-always excellent technical support, owing to which not a single of my technical issues stays unresolvable. Special thanks to Kaicong Sun, Roman Krawtschenko, Zhe Wang and Seyyed Mahdi Najmabadi for joining me on the research path. I am grateful for our productive cooperation and co-author publications.

I want to thank Prof. Dr.-Ing. Andrés Bruhn for his enthusiastic course “Correspondence Problems in Computer Vision” and being the chair of my committee. I made the right decision by taking part in his course back in 2016 whose valuable knowledge allowed me to make the very first step into the research topic. I am also grateful to Prof. Dr. David Taubman for his knowledgeable advice and cooperation. Without his valuable input, I could not go forward with the 4D light field compression. I very much appreciate Prof. Dr. Hassan

Rabah for being my second referee and Jun.-Prof. Dr. rer. nat. Benjamin Uekermann for his time and effort in joining the committee.

Last but most importantly, I would like to express my deepest gratitude to my parent, my wife, Anh, and my little precious, Hannah for their unconditional love and support. They gave me the courage to move forward and were always there to cheer me up. This dissertation is dedicated to them.

Contents

List of Abbreviations

AAW	Angular Attention Weight
ADMM	Alternating Direction Method of Multipliers
AP	Angular Patch
ASIC	Application-Specific Integrated Circuit
ASR	Angular Super-Resolution
ASSR	Angular-Spatial Super-Resolution
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DWT	Discrete Wavelet Transform
EPI	Epipolar Image
EVRN	EPI Volume-based Refinement Network
FBN	Feature Bottle-neck Layer
FPGA	Field Programmable Gate Array
GPU	Graphic Processing Unit
HDL	Hardware Description Language
HLS	High Level Synthesis
LF	Light field
LFSR	Light-field Super-Resolution
MISR	Multi-Image Super-Resolution
MWD	Motion-compensated Wavelet Decomposition
MRF	Markov Random Field
NLTV	Non Local Total Variation
PASR	Preliminary Angular Super-Resolution
PCA	Principal Component Analysis
PreLU	Parametric Rectified Linear Unit
PSNR	Peak Signal-to-Noise Ratio
PSSR	Preliminary Spatial Super-Resolution
SAI	Sub-Aperture Image
SAW	Spatial Attention Weight

SFE	Shallow Feature Extraction
SISR	Single Image Super-Resolution
SSIM	Structural Similarity Index Measure
SSR	Spatial Super-Resolution
TV	Total Variation
WMF	Weighted Median Filtering

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 Introduction

Light field refers to the concept of acquiring a complete description of light rays emitted from a scene and traverse in space. It can be well parameterized by the 7D plenoptic function $\boldsymbol{L}(p_x, p_y, p_z, \theta, \phi, \lambda, t)$ [1] which returns the radiance (intensity) of a line beam observed at a point (p_x, p_y, p_z) in space, along a gazing direction (θ, ϕ), at a time t and wavelength λ. In practice, it is of interest to capture only a snapshot of the function (at a fixed time) and simplify the spectral information by using only 3 color components (i.e. red, green, blue). By removing the time and wavelength parameters, we are left with a 5-parameters function, also known as 5D plenoptic function [2], which allows us to describe the intensity of any light ray in 3D space regardless the viewpoint position and the change of light direction. One more parameter can be omitted from the function, if one assumes that the direction of a light ray is unchanged and considers only the light intensity being visible at a specific position, i.e. placing of cameras. This is also the common setup of light field imaging which results in a 4D light field [2] or Lumigraph [3]. Under the two plane parameterization [2], each ray is indexed with a 4D coordinate by its intersection with two parallel planes, as depicted in Figure 1.1a.

$$\boldsymbol{L} : \mathit{\Omega} \times \mathit{\Pi} \rightarrow \mathbb{R}, \qquad (\mathbf{z}, \boldsymbol{\theta}) \rightarrow \boldsymbol{L}(\mathbf{z}, \boldsymbol{\theta}) \tag{1.1}$$

where $\mathbf{z} = (x, y)^T$ and $\boldsymbol{\theta} = (\rho, \tau)^T$ denote coordinate pairs in the spatial plane $\mathit{\Omega} \subset \mathbb{R}^2$ and in the directional plane $\mathit{\Pi} \subset \mathbb{R}^2$. In practice, the value of this function can be a

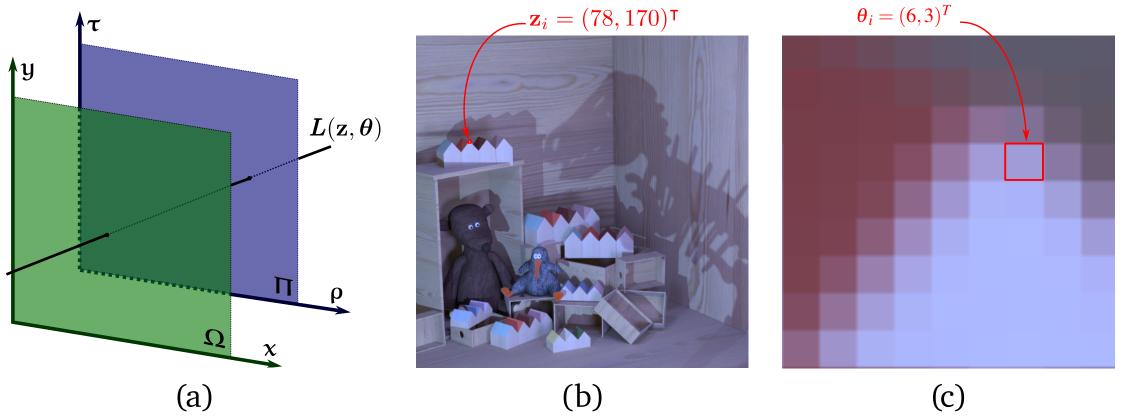


Figure 1.1: Light field representation and acquisition. (a) Two-plane parameterisation. (b) A sub-aperture image at angular location $\boldsymbol{\theta}_i = (6, 3)^T$. (c) An angular path at spatial location $\mathbf{z}_i = (78, 170)^T$

vector of 3 color components (i.e. RGB color light field) and the two planes are discretized by the sampling rate of capturing devices (i.e. sensor size, number of cameras,...).

Given a 4D light field $\boldsymbol{L}$, by fixing either the direction index $\boldsymbol{\theta} = \boldsymbol{\theta}_i$ or spatial index $\mathbf{z} = \mathbf{z}_i$, one can obtain a sub-aperture image $\boldsymbol{L}(\mathbf{z}, \boldsymbol{\theta}_i)$ or an angular patch $\boldsymbol{L}(\mathbf{z}_j, \boldsymbol{\theta})$. Figure 1.1 (b),(c) are examples of a sub-aperture image and an angular patch extracted from a 4D synthetic light field image, *'dino'* [4]. The resolution of this light field is 512×512×9×9, where 512×512 is spatial resolution and 9×9 is angular resolution. The angular patch consists of pixels, each at the same spatial location $\mathbf{z}_i = (78, 170)^T$ on sub-apertures.

The main advantage of a light field image over the conventional image is its high dimensional imaging content, as a light field image contains both directional information and spatial information instead of only spatial information as in conventional image [5]. This rich-content property of light field brings a great benefit to numerous applications such as in autonomous systems [6], virtual reality [7], 3D television [8]. However, this benefit also comes with a cost of computational resources. Processing 4D LF images requires more memory bandwidth, computing power and runtime than the conventional 2D image. Essential LF processing tasks such as disparity estimation, high-resolution reconstruction and compression generally involves global optimization problem in which the solution can only be iteratively calculated. Taking into account the large volume of image-like data and the intensive floating-point computation of iterative solvers, Graphics Processing Units (GPUs) obviously becomes a dominant platform for offloading and accelerating the LF processing tasks. This dissertation investigates the high-performance processing of 4D LF image as a combination of computing time and result quality. Particularly, we study the three essential LF processing tasks, i.e., disparity estimation, super-resolution, and compression. For each task, we look for an approach in which both the algorithmic aspects for high quality output and GPU-acceleration aspect for fast computation are considered in order to achieve a high-performance outcome.

1.2 4D Light field Acquisition and Visualization

In general, light field acquisitions can be categorized in to three main classes: multi-sensor capturing [9], time-sequential capturing [10] and multiplexed imaging [11–13]. The multi-sensor capturing approach requires an array of image sensors distributed on a planar or spherical surface to simultaneously capture light field samples from different viewpoints. The time-sequential capturing approach, on the other side, uses a single image sensor to capture multiple samples of light field through multiple exposures. The typical approach uses a sensor mounted on a mechanical gantry to measure the light field at different possitions [10]. The multiplexed imaging encodes high dimensional light field into a 2D sensor plane, by multiplexing the angular domain into the spatial domain. One popular example of this acquisition approach is plenoptic camera [11–13] in which a microlens array is placed in between a main-lens and an image sensor. Each acquisition method has its own advantages and disadvantages. Multi-sensor capturing approach is generally more expensive but allows to capture very high spatial resolution of a dynamic screen. The time-sequential capturing approach is inexpensive and can capture very high spatial resolution but suffers from very long capturing time which makes it less preferable for capturing a dynamic scenes. Multiplexed imaging approach is inexpensive and can handle dynamic scene but produces low spatial resolution images. In addition, all acquisition approaches impose a trade-off between angular resolution and either spatial resolution or temporal resolution, i.e.: reducing the cost of the sensor by using low-resolution sensors in exchange for increasing the number of sensors for higher angular resolution; increasing capturing time in order to capture dense angular samples of the scene; increasing the number of microlenses in order to increase the spatial resolution but at the same time reducing the number of angular samples.

Seven published light field datasets, which are widely used in the literature, are listed in Table 1.1. These dataset are employed in this dissertation for the evaluation of the proposed approaches and the comparison to the related works. The datasets are categorized into two groups: synthetic scenes and real-world scenes. The synthetic scene datasets [4, 14, 15] are generated by 3D object models and Blender software. This type of datasets includes the disparity values and can be used for the quantitative evaluation of disparity

Table 1.1: Summary of published synthetic and real-world 4DLF dataset

Datasets	Type	Angular Res.	Spatial Res.	Size
HCI13[14]	synthetic	9×9	misc	7
HCI17[4]	synthetic	9×9	512×512	28
InSyn[15]	synthetic	9×9	512×512	92
StGantry[19]	real-world	17×17	misc	13
StLytro[16]	real-world	14×14	375×541	350
InLytro[17]	real-world	15×15	434×625	36
EPFL[18]	real-world	15×15	434×625	118

estimation approaches, as in chapter 2 and 3. The real-world datasets are captured by Illum cameras [16–18] and a gantry setup [19].

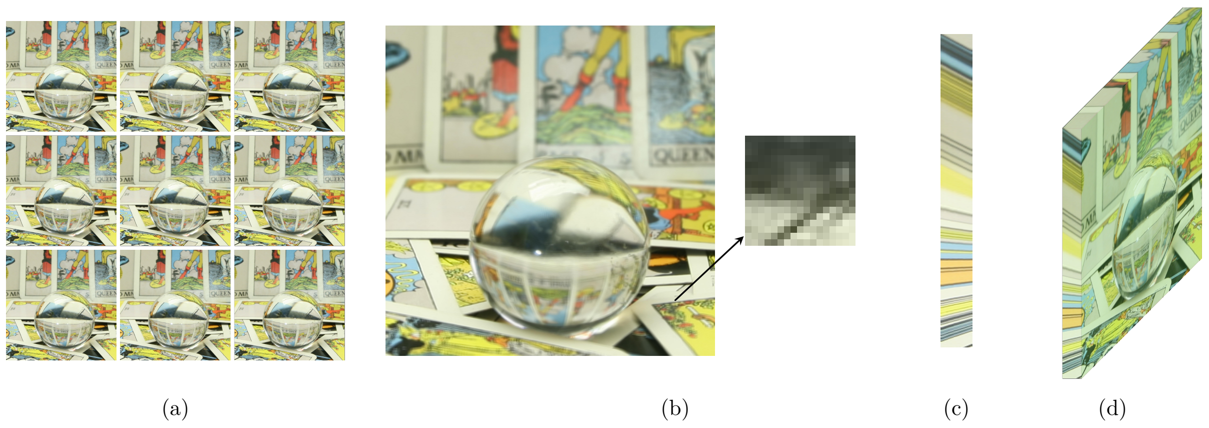


Figure 1.2: 4DLF visualization. (a) 2D array of sub-aperture images; (b) 2D array of angular patches; (c) 2D EPI; (d) EPI volume.

The four dimension basis of light field makes it difficult for visualization and analysis. Therefore, 4DLF is normally transformed into a set of lower dimensional data, such as 2D images or 3D volume. The 2D representation includes sub-aperture image (SAI) [20, 21], angular patch (AP) [22, 23], and epipolar image (EPI) [24, 25]. The 3D representation is also denoted as EPI volume [26, 27]. Figure 1.2 illustrates the variations of 4DLF representation.

1.3 Related Works

1.3.1 Disparity Estimation

Disparity estimation in 4D light field image processing may also refer to multi-view image registration which aims to solve a correspondence matching problem. The output is a displacement map $\boldsymbol{\omega}$ ($\boldsymbol{\omega} : \Omega \longrightarrow \mathbb{R}$) that represents the correspondence of pixels between sub-aperture images. Suppose that the scene contains only Lambertian surfaces, the following equality is the well-known property of a 4D light field [1]

$$\boldsymbol{L}(\mathbf{z}, \boldsymbol{\theta}_0) = \boldsymbol{L}\big(\mathbf{z} + \boldsymbol{\theta}_k \boldsymbol{\omega}, \boldsymbol{\theta}_k\big) \tag{1.2}$$

$\boldsymbol{\theta}_0 \in \Pi$ presents a reference view position for which $\boldsymbol{\omega}$ is calculated and $\boldsymbol{\theta}_k \in \Pi$ is an arbitrary view position. This equality shows that the two corresponding pixels, the pixel at the position $\mathbf{z}$ in the view $\boldsymbol{\theta}_0$ and the pixel at the position $\mathbf{z} + \boldsymbol{\theta}_k\boldsymbol{\omega}$ in the view $\boldsymbol{\theta}_k$, have the same intensity value. This equation, also known as a brightness constancy assumption, is due to the fact that the light beam which contacts these pixels' locations comes from the same 3D point in the captured scene. Knowing $\boldsymbol{\omega}$, along with camera poses information, one can derived depth value associated with each 3D point in the captured scene. Therefore, the two terms disparity map and depth map are used interchangeably in the literature.

In this section, a literature review of disparity estimation and recent efforts on accelerating light field image processing are discussed. We limit our discussion on approaches that work on 4D light field data and categorizes them into two groups: traditional and learning approach. The traditional approach refers to the one which estimates a disparity map directly from an input LF image, normally using a local or global optimization scheme. Learning approach, on the opposite side, predicts the disparity map by learning a large amount of LF data with a similar setup.

Most of the conventional approaches formulate the disparity estimation task as a multi-label optimation problem in which the disparity range is discretized by a step size to form a label set [23, 28–34]. A cost volume is then constructed by associating each label assignment with some costs based on the manipulation of LF image properties. Combining with priors, the final disparity map is computed by regularization in a Markov Random Field (MRF) framework. Jeon et al. [28] used the sum of absolute and gradient differences as the label costs and proposed to use the phase-shift theorem to achieve better accuracy in narrow

[1] Without the loss of generality, we shift the coordinate of Π by $\boldsymbol{\theta}_0$ which leads to $\boldsymbol{\theta}_0 = \begin{bmatrix}0 & 0\end{bmatrix}^T$ and $\boldsymbol{\theta}_k = \boldsymbol{\theta}_k - \boldsymbol{\theta}_0$.

baseline LF images. Tao et al. [29] suggest combining defocus and correspondence cue for data costs which are then improved in [30] by adding occlusion response as a regularization term. In [31], Neri et al. proposed to estimate the disparity map at various resolutions to reduce the accuracy loss in flat surface regions. The author mitigated the computational complexity by approximately solving a local minimization of functional. In [32], Si et al. construct a cost volume using photo consistency constraint and regularized with a second-order smooth term. Lee et al. [34] employed the Lambertian assumption and gradient constraint and proposed foreground-background separation strategy to build a cost volume. Anisimov et al. [33] used the census transform and Hamming distance to define label costs which are then aggregated using Semi-Global Matching (SGM) principle. Williem et al. [23] proposed two novel data costs, constrained angular entropy cost and adaptive defocus cost which are showed robust again both noise and occluder.

Due to the flexibility in constructing label costs, multi-label optimization approaches are generally robust against outliers and can provide a very accurate disparity map. However, the common drawback of this type of approach is the high demand for memory and computation in order to achieve a global optimum result. A general cost volume has the resolution of $H \times W \times D \times C$, where $H \times W$ is the resolution of the disparity map, D and C are the numbers of disparity labels and label costs respectively. The accuracy highly depends on the size of the label set which is discretized from an assumed disparity range. Increasing the size to cover a finer disparity step would result in more memory and computation requirements. Other approaches formulate the disparity map as a continuous function and solve a variational optimization problem [24, 35]. Wanner et al. [24] initialized disparity maps by local estimation of slope in Epiponar Plan Image (EPI). These maps wer then used for weighting a data fidelity term in a global optimization scheme. Strecke et al. [35] showed that depth map and normal map have a strong relationship and proposed to jointly compute these two maps from a 4D light field.

Given enough training data, learning approaches can provide a higher accuracy compared to traditional approaches [36]. In [37], Lou et al. trained a Convolution Neural Network (CNN) to classify EPI-patch. The network predicted a confident cost to assign a depth label to each pair of EPI-patch, vertical and horizontal. The cost volume was then regularized in an MRF model. In [38], Heber et al. proposed a U-shaped network including encoding and decoding parts which compute the disparity for each EPI line in the input EPI image. The network is limited to processing only one spatial direction and results in a striking artifact in the respective direction. This problem was then corrected in its enhanced version [26] employed 3D convolutional operator and predict the disparity maps from 3D EPI volume. VommaNet [39] proposed an end-to-end CNN approach that retrieves multi-scale features and generates accurate disparity for reflective and texture-less regions. Epinet [25] uses a fully convolutional neural network to predict a disparity map directly from an input 4D light field. The network took in four angular sets of sub-aperture images (i.e. horizontal, vertical, diagonal and reverse diagonal) and output the disparity map in the central angular position. Epi-shift [40] proposed a novel CNN-based approach that can deal with a wide baseline light field while retaining a small receptive field.

Besides the CNN-based approaches that relied on the computation capability of deep learning frameworks, the number of existing works on accelerating disparity estimation is quite limited. In [41], Yuttaknonkit et al. optimized the memory access pattern in order to deal with the memory bottleneck in rendering and depth extraction applications. They worked on the plenoptic 2.0 image [13] in which the depth extraction process is much simpler than 4D light field and only involves a local search of pixel patch on microlens images. Qin et al. [42] and Ivan et al. [43] presented GPU acceleration architecture for cost-volume based approaches. The cost volume in the first work is generated by simply shifting the sub-aperture images toward the center angular and calculating the maximum difference. This cost volume is less advanced compared to the one in [43] that employed the matching cost from [23]. Both works skipped the global minimum solution and made use of the winner-take-all algorithm to simplify the computation.

1.3.2 Light Field Super-resolution

The objective of LF image super-resolution is to reconstruct a higher resolution LF image $\boldsymbol{L}^h(\mathbf{z}, \boldsymbol{\theta})$ given a lower resolution one $\boldsymbol{L}^l(\mathbf{z}, \boldsymbol{\theta})$. Let $(h_\tau, h_\rho, h_y, h_x)$ and $(l_\tau, l_\rho, l_y, l_x)$ be the resolution of $\boldsymbol{L}^h$ and $\boldsymbol{L}^l$ respectively, the upscaling ratio is then $(\zeta_\tau, \zeta_\rho, \zeta_y, \zeta_x) \equiv (\frac{h_\tau}{l_\tau}, \frac{h_\rho}{l_\rho}, \frac{h_y}{l_y}, \frac{h_x}{l_x})$. Where ζ_τ is angular vertical scale, ζ_ρ is angular horizontal scale, ζ_y is spatial vertical scale and ζ_x is spatial horizontal scale. In the literature, light field super-resolution problem generally deals with an equally scaling factor for both horizontal and vertical dimension, i.e., ζ_x=ζ_y=$\zeta_\mathbf{z}$ and ζ_τ=ζ_ρ=$\zeta_{\boldsymbol{\theta}}$. Depends on the values of directional scaling factor ($\zeta_{\boldsymbol{\theta}}$) and spatial scaling factor($\zeta_\mathbf{z}$), there are different light field super-resolution problems: spatial super-resolution (SSR), angular super-resolution (ASR), and angular-spatial super-resolution (ASSR).

In general, previous works can be categorized into two groups: optimization-based approaches and learning-based approaches. In optimization-based approaches, LF super resolution is formulated as an optimization problem which typically consists of a data fidelity term directly composed from input LF image and a regularization term based on known priors. There are two main types of data terms that are used in the literature. One of them penalizes the coherence between low and high-resolution LF image pairs [44–47].

The other enforces the Lambertian consistency across the directional dimension by warping SAIs from different view angles using pre-computed disparity maps [24,47,48]. Compared to the data fidelity term, the choices of regularization terms are more diverse. Each work proposed to use a different prior in order to achieve a better output quality and with a feasible computation effort, i.e., total variation (TV) [24], bilateral TV[48], Markov Random Field (MRF) [45], Gaussian [44], Graph-based [49], sparsity [46]. In [45], Bishop et al. studied an explicit image formation model that characterizes the light field imaging process by spatially-variant point spread functions (PSFs). The PSFs were derived under Gaussian optics assumptions and employed in a Bayesian framework for super-resolution. In [44], Mitra et al. showed that 4-D patches of different disparities have different intrinsic dimensions and proposed to learn a Gaussian prior for each quantized disparity value. These priors were then employed to inference high-resolution 4-D patches under the Maximum a posterior (MAP) criterion. LF super-resolution was modeled as a continuous optimization problem using a variational framework in [24]. Disparity maps were extracted by local estimation of pixel-wise slope in EPI. The data fidelity term was constructed by warping surrounding views with the estimated disparity maps and masking with occlusion maps, while total variation was used for regularization. In [48], Tran et al. treated LF super-resolution as a multi-frame super-resolution problem in which degradation process is modeled by three operators: down-sampling, blurring, and warping. A variational optimization approach was employed to estimate disparity maps used by the warping operator and bilateral TV was employed as an image prior. In [46], Alain et al. proposed a patch-based super-resolution approach making use of a 5D transform filter that consists of 2D DCT transform, 2D shape-adaptive DCT and 1D haar wavelet. By a proper selection of 5D patches, a transformed signal exposes a high degree of sparsity which can be employed as a prior to regularise a L^2 data term. In [49], Rossi et al. proposed an approach which couples two data-terms with a graph-based regularizer. A graph-based prior regularizes high-resolution SAIs by enforcing the geometric light field structure. Block matching was employed in their work for the estimation of disparity values and the construction of the graph map.

Concerning Learning-based approaches, the earliest work which applies deep-learning for reconstructing high-resolution LF is [50]. The authors employed the SISR approach from [51] for SSR and proposed a CNN-based solution for ASR in which novel views depending on its position will be synthesized from a vertical pair, a horizontal pair, or four neighbors. An improved version was described in [52] where SISR was applied to each SAI separately and learning variables were shared in the ASR network. In [53], Kalantari et al. proposed to generate novel SAIs by exploiting disparity information in a two-stage CNN. The first stage predicts disparity maps from a pre-computed cost-volume and the second stage synthesizes novel views from input images that are warped using predicted disparity maps. In [54], a patch-based approach that employes linear subspace projection was presented. A linear mapping function between low and high subspaces of low and high LF patch volume was learned from a training dataset and applied to new LF images. The authors used block matching to find matched 2D patches and extract aligned patch volumes. Principal component analysis (PCA) was then employed to reduce patch volumes' dimensions and project them into subspaces. The mapping function was computed in the form of a l_2-norm regularized least square problem. Fan et al. [55] proposed a two-stage approach for SSR. In the first state, each SAI was upscaled using the SISR approach from VDSR [56]. The output was then registered to a reference SAI by locally searching similar patches. Both reference image and registered images were fed into a CNN network in the second stage to reconstruct high-resolution SAI at the reference position. Similarly, Yuan et al. [57] proposed another two-stage approach. In the first stage, EDSR [58] was employed to super-resolute SAIs. In the second stage, the output of SISR was then enhanced by a refinement CNN which relies on 2D EPI. In [22], Gul et al. proposed an approach targeting lenslet images captured by plenoptic cameras [11, 12]. Microlens image patches were used as input to two separate networks, i.e., one for ASR and the other for SSR. However, the fully-connected layers employed in these networks limited its application to a certain angular resolution. Wang et al. [59] developed a bidirectional recurrent CNN approach for spatially upsampling 4D LF images. They employed multi-scale fusion layers for future extraction to accumulate contextual information. Two networks for vertical and horizontal image stacks were learned separately and a stack generalization technique was employed to obtain a complete set of images. To fully exploit the 4D structure of LF images, Yeung et al. [60] proposed to apply 4D convolution for SSR. The 4D convolution function was implemented as spatial-angular separable convolution which allows extracting feature maps from both spatial and angular domains. In [61], Zhang et al. proposed a residual CNN-based approach for reconstructing LF images with higher spatial resolution. The network takes in image stacks from four different angles and predicts a high-resolution image at the center position. According to the difference in angular position, it requires six different networks for a full reconstruction of the 4D LF image.

Despite its importance the acceleration of 4DLF super-resolution has drawn quite limited attention in the literature. Most of the previous works focused only on the reconstruction quality aspect and ignore the timing aspect. The proposed approaches generally consist of computational expensive operations, e.g. graph processing [49] or 5D filtering [46]. One of the efforts to accelerate the super-resolution task was discussed in [24], whose approach resembles the classical SR model presented in [62]. However, this work merely used GPU for experimental purpose. It lacks of detailed discussion on the arrangement of computing tasks and dealing with high dimensional LF data. Although multi-frame super-resolution approaches [63, 64] can also be applied for 4DLF, considering the representation of 2D array of SAIs, 4DLF has its own properties which could be employed for better modeling and regularization.

1.3.3 Light Field Image Compression

Along with the increasing popularity of light field photography, the effective coding of light field images has become an imperative challenge. Recent works on light field image coding mainly focused on High Efficiency Video Coding (HEVC) framework, either proposing new prediction tools [65–67] or directly applying HEVC to encode light field data [68, 69]. Liu et al. [68] proposed to reorder sub-aperture images as a pseudo video sequence and then encode it using HEVC. In [69], Perra et al. applied a similar procedure but instead of using sub-aperture images, they proposed to tile raw light field image into non-overlaping sub-images. Ricardo et al. [65] proposed two prediction tools to HEVC framework that base on the locally linear embedding and self-similarity compensated prediction. Li et al. [66] worked on an adapted version of bi-directional inter-frame prediction to intra prediction for coding raw light field images. In [70], Petri et al. proposed a coding scheme that utilizes provided disparity map from Lytro software to segment the sub-aperture images and losslessly encode them.

1.4 Contributions

The 4D parameterization of light field enables a seamless processing workflow of Light field image. Images captured by different techniques, i.e. plenoptic camera [11, 12], camera array [9], gantry setup [10], are converted into the uniform 4D representation for which further processing tasks are effectively performed. Figure 1.3 presents the typical LF image processing tasks studied in our work. Regardless of acquisition methods, the raw images are processed and assembled into a 4DLF format through a pre-processing step involving important tasks such as de-mosaicing, calibration, rectification, hot-pixel detection [71]. On top of the 4DLF data, various processing tasks can be performed, which include disparity estimation, high-resolution reconstruction, and compression as depicted in Figure 1.3.

Disparity estimation is the most essential task in which the disparity maps are extracted from the 4DLF data. The disparity maps holds important information describing the

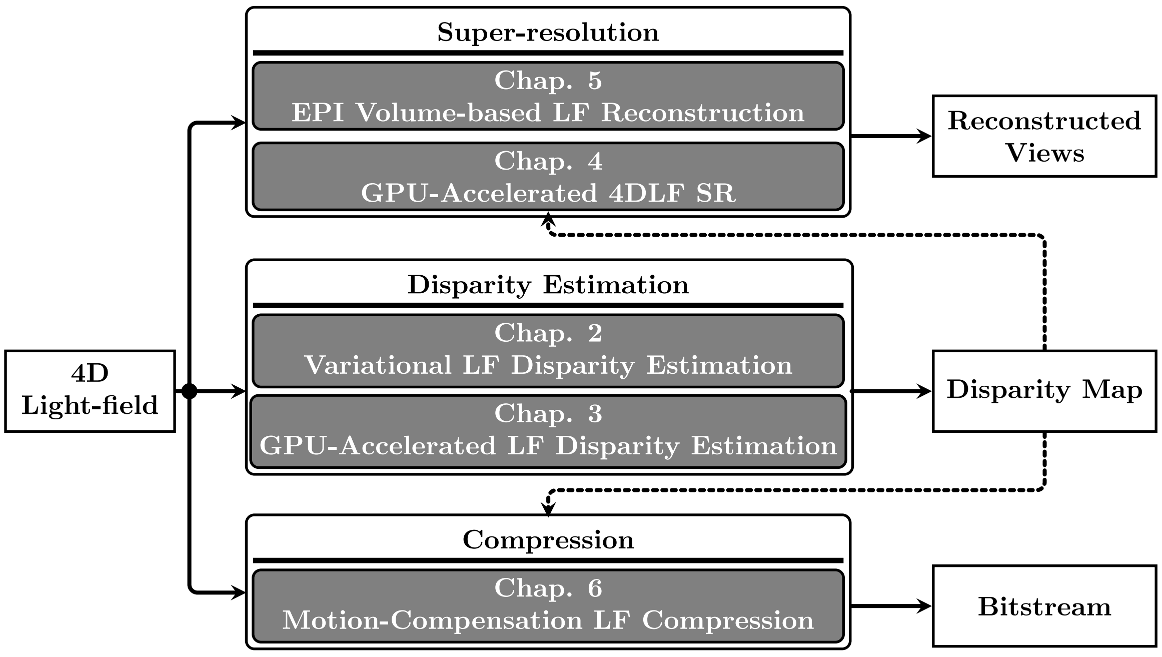


Figure 1.3: Light field image processing workflow

3D structure of the captured scene and are employed in the other processing tasks, i.e., super-resolution and compression. For 4DLF disparity estimation task, we applied in the first step the variational computation framework from optical flow literature to estimate a 4DLF disparity map which is then enhanced in a post-processing step. The proposed approach greatly benefits from the intrinsic pixel precision of variational formulation and the effectiveness of weighted median filtering as a post-processing technique to achieve a highly accurate solution. A fast parallel computation on GPUs are presented for alleviating the high computational demand of solving the optimization problem.

Given the the disparity maps, sub-aperture images can be well approximated from each other by applying warping technique. Due to the fact that the warped sub-aperture images expose a high degree of intensity similarity, we proposed an invertible motion-compensated wavelet decomposition scheme for 4D LF compression. The decomposition scheme effectively removes redundant information across multiple perspective images and produces a set of the high-/low-pass sub-band views which are then encoded by JPEG2000 encoder. The proposed approach supports both lossless and lossy compression scenarios.

The acquisition of 4DLF assembling high-resolution information in both spatial and angular domain remains a technology challenge which encourages intensive researches on light field super-resolution (LFSR). LFSR aims for solving the ill-posed inverse problem in which high-resolution 4D LF is reconstructed from a given low-resolution 4DLF. The increment in the resolution is considered for spatial domain, angular domain, and angular-spatial domain which results in different super-resolution problems namely spatial SR, angular SR, and angular-spatial SR, respectively. For spatial SR, we proposed a GPU-accelerated framework which assembles mixed Gaussian-Impulse noise model and weighted regularization. As a part of the degradation model, the disparity maps are employed to form data fidelity terms considering the mix-noise condition from a statistical perspective. The framework generalizes previous multi-image super-resolution approaches considering Total-variation (TV) based regularization.

As an attempt to solve the three LFSR problems, we proposed a deep-learning-based approach in which the SR reconstruction is carried over the two processing stages. The first stage preliminarily upsampling the size of 4D LF in spatial, angular or angular-spatial domains. A novel refinement convolutional neural network based on 3D EPI volume is proposed in the second stage to enhance the quality of the reconstructed 4DLF. Through 3D convolutional operations and efficient deep-learning architectures, angular and spatial information presented in the 3D EPI volume structure is effectively exploited to reconstruct high-frequency details. As a result, the reconstructed high-resolution light field demonstrates a balanced performance distribution across all perspective images and presents superior visual quality compared to the previous works.

In summary, the main contributions of our research are as follows:

- 4DLF disparity estimation approach from flow-field perspective and accuracy enhancement by post-processing are presented in Chapter 2. Paralleled implementation of the proposed approach on GPU platforms is presented in Chapter 3.
- Optimization-based spatial super-resolution framework for 4DLF which couples $\ell^1 - \ell^2$ data fidelity terms and weighted regularization. Alternating direction method of multiplier is applied for solving the optimization problem and transforming it into a simpler form which is suitable for GPU acceleration. The iterative solver are effectively realized on GPU platform resulting in a significant performance boost, Chapter 4.
- A deep learning-based approach based on 3D epipolar image (EPI) for the reconstruction of high-resolution 4DLF. Through a 2-stage super-resolution framework, the proposed approach effectively addresses various LFSR problems, i.e., spatial SR, angular SR, and angular-spatial SR. While the first stage provides flexible options to up-sample EPI volume to the desired resolution, the second stage, which consists of a novel EPI volume-based refinement network (EVRN), substantially enhances the quality of the high-resolution EPI volume, Chapter 5.

- 4DLF compression approach based on motion-compensated wavelet decomposition and its fast parallel computation on GPUs are presented in Chapter 6.

1.5 Organization

The rest of this dissertation is organized as follows.

- Chapter 2 presents an approach for estimating the 4DLF disparity map which assembles variational optimization framework and post-processing technique. A detailed solving strategy and numerical computation are also discussed.
- Chapter 3 presents the fast parallel computation of 4DLF disparity estimation on GPUs. It also discusses a detailed analysis of the OpenCL implementation as well as an extensive comparison to related approaches.
- Chapter 4 presents an optimization-based approach for reconstructing spatially HR image from low-resolution 4DLF degraded by mixed Gaussian-Impulse noise. Iterative solvers and their accelerated realization on GPUs are also discussed.
- Chapter 5 presents a novel deep learning-based approach for spatial, angular, and angular-spatial super-resolution of 4DLF. An extensive evaluation and comparison to related methods are also discussed.
- Chapter 6 presents a method to compress 4DLF image based on a motion-compensated wavelet lifting scheme and the application of JPEG2000 encoder. The GPU acceleration of the decomposition scheme is also discussed in this chapter.
- Chapter 7 concludes the thesis by summarizing the core contributions and discussing future research directions.

Chapter 2

Variational 4D Light Field Disparity Estimation

This chapter discusses the proposed approach to estimate the 4DLF disparity maps. We follow the optical flow literature to formulate the disparity estimation problem in the form of a variational optimization problem. Weighted median filtering is employed as a post-processing technique to enhance the accuracy of the estimated 4DLF disparity map. The chapter starts with a brief introduction to the 4DLF disparity estimation problem and essential notations. The variational optimization problem is presented in the next section, which discusses some configurations of data fidelity term and smoothness term. The optimization strategy to solve the minimization problem Sec. 2.3, while its numerical computation is in Sec. 2.4. Experimental results compared to commercial software and contemporary approaches on both synthetic and real-world datasets are reported in Sec. 2.5. Parts of the results of this chapter have been published in [72].

2.1 Problem Setup and Notation

A disparity map (also referred to as a displacement field) is a term commonly used in stereo vision applications to describe the correspondent matching output [73]. A disparity map represents pixel-wise motion from one image to the other and is usually defined in the form of a 2-tuple mapping function $M(x, y) = \begin{bmatrix} u(x, y) & v(x, y) \end{bmatrix}^\top$, where functions u and v, $u, v : \mathbb{N}^2 \rightarrow \mathbb{R}$, denote the pixel shifts in x direction and y direction, respectively. Given an image pair consisting of a left image I_l and a right image I_r, the correspondent matching property of the disparity map is described by the following equality,

$$\Phi\big(I_l(x,y)\big) = \Phi\Big(I_r\big(x + u(x,y), y + v(x,y)\big)\Big), \tag{2.1}$$

where, Φ is a feature function that returns a feature value at a pixel location (x, y). Some examples of Φ are intensity value, gradient value [74]. Eq. 2.1 represents a constancy assumption which states that the feature values of two corresponding pixels, i.e., (x, y) in I_l and $\big(x + u(x,y), y + u(x,y)\big)$ in I_r, are equal. The two motion functions u and v are computed with regard to the left image I_l.

As discussed in the previous chapter, one typical representation of a 4D LF, i.e., $\boldsymbol{L}(\mathbf{z}, \boldsymbol{\theta})$, is a 2D arrangement of its sub-aperture images. This representation brings us closer to the context of multi-view stereo vision which deals with multiple images captured from various camera positions. However the 4D LF imposes a constraint on the placements of the cameras leading to much simpler configuration. Since the sampling in the directional domain Π is assumed to be regular, it can be pictured as placing cameras on a flat surface with equal spacing. For defining a disparity map in the case of 4D LF, let us first re-index the directional coordinate $\boldsymbol{\theta}$ by shifting its origin to the location of a reference SAI, i.e. $\boldsymbol{L}(\mathbf{z}, \boldsymbol{\theta}_0)$, where $\boldsymbol{\theta}_o$ is the directional index of the reference SAI. We denote the disparity map with regard to $\boldsymbol{L}(\mathbf{z}, \boldsymbol{\theta}_0)$ as $\boldsymbol{\omega}(\mathbf{z})$. Let us define $M_i(\mathbf{z}) = \begin{bmatrix} u_i(\mathbf{z}) & v_i(\mathbf{z}) \end{bmatrix}^\top$ is the motion mapping functions between the reference SAI and the SAI located at $\boldsymbol{\theta}_i$. The relationship between M_i and $\boldsymbol{\omega}$ is derived as follows,

$$M_i(\mathbf{z}) = \begin{bmatrix} u_i(\mathbf{z}) \\ v_i(\mathbf{z}) \end{bmatrix} = \begin{bmatrix} \kappa & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \rho_i - \rho_0 \\ \tau_i - \tau_0 \end{bmatrix} \boldsymbol{\omega}(\mathbf{z}) = K(\boldsymbol{\theta}_i - \boldsymbol{\theta}_0)\boldsymbol{\omega}(\mathbf{z}), \tag{2.2}$$

where κ denotes a scalar constant compensating for the difference between horizontal and vertical units (i.e., due to the non-square pixel, difference in horizontal/vertical spacing of cameras), $\boldsymbol{\theta}_* = [\rho_*, \tau_*]^\top$, $K = \begin{bmatrix} \kappa & 0 \\ 0 & 1 \end{bmatrix}$. By shifting the directional coordinate (i.e., $\boldsymbol{\theta}_0 = [0,0]^\top$) and absorbing K into $\boldsymbol{\theta}$, we come to the final disparity map relation,

$$M_i(\mathbf{z}) = \boldsymbol{\theta}_i \boldsymbol{\omega}(\mathbf{z}) \tag{2.3}$$

To further simplify the notation, let us rewrite $\boldsymbol{L}_i(\mathbf{z}) = \boldsymbol{L}(\mathbf{z}, \boldsymbol{\theta}_i)$ signifying the SAI at location $\boldsymbol{\theta}_i$, and write $\boldsymbol{\omega}$ instead of $\boldsymbol{\omega}(\mathbf{z})$. The equality in Eq. 2.1 can be rewritten for a pair of SAIs as,

$$\boldsymbol{L}_0(\mathbf{z}) = \boldsymbol{L}_i(\mathbf{z} + \boldsymbol{\theta}_i \boldsymbol{\omega}). \tag{2.4}$$

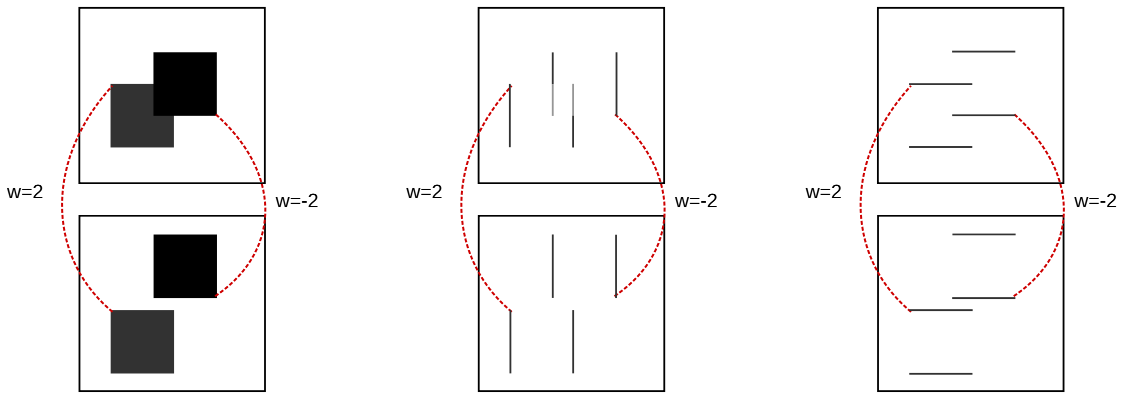


Figure 2.1: Illustration of photo constancy and gradient constancy assumptions of two adjacent SAIs

2.2 Variational Disparity Estimation

This section discusses the proposed approach for light field disparity estimation. We follow optical flow literature [74, 75] to define disparity estimation as the following variational optimization problem

$$\widetilde{\boldsymbol{\omega}} = \underset{\boldsymbol{\omega}}{\arg\min} \int_{\Omega} D(\boldsymbol{\omega}) + S(\boldsymbol{\omega})\, d\mathbf{z}. \tag{2.5}$$

The disparity map is formulated as a continuous function ($\boldsymbol{\omega} : \Omega \longrightarrow \mathbb{R}$). $D(\boldsymbol{\omega})$ and $S(\boldsymbol{\omega})$ are data fidelity term and regularization term respectively. The data fidelity term penalizes the constancy assumptions of 4DLF data, while the regularization term guarantees a reasonable solution by enforcing known priors.

2.2.1 Data Term

Based on 4D LF properties, the data term is constructed with an assumption on constancy. The two common constancy assumptions which were employed in the literature are brightness (photo) constancy and gradient constancy [74,76]. The first assumes that the intensity value of corresponding pixels is unchanged. Given a pixel $\mathbf{z}_j$ on the reference view $\boldsymbol{\theta}_0$, its corresponding pixel on the view $\boldsymbol{\theta}_i$, follow Eq. 2.1, is $\mathbf{z}_j + \boldsymbol{\theta}_i\boldsymbol{\omega}$. The brightness constancy assumption reads that $\boldsymbol{L}_0(\mathbf{z}) = \boldsymbol{L}_i(\mathbf{z} + \boldsymbol{\theta}_i\boldsymbol{\omega}), \quad \forall \boldsymbol{\theta}_i \in \Pi$. To plot this assumption into our optimization framework, we define the error function in intensity between corresponding pixels as follows

$$F_{g,i}(\boldsymbol{\omega}) = ||\boldsymbol{L}_0(\mathbf{z}) - \boldsymbol{L}_i(\mathbf{z} + \boldsymbol{\theta}_i\boldsymbol{\omega})||_2^2 \tag{2.6}$$

The gradient constancy assumption implies the equality of the local changes of two corresponding pixels,

$$\begin{aligned} \nabla \boldsymbol{L}_0(\mathbf{z}) &= \nabla \boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega}) \\ \leftrightarrow \begin{bmatrix} \frac{\partial}{\partial x}\boldsymbol{L}_0(\mathbf{z}) \\ \frac{\partial}{\partial y}\boldsymbol{L}_0(\mathbf{z}) \end{bmatrix} &= \begin{bmatrix} \frac{\partial}{\partial x}\boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega}) \\ \frac{\partial}{\partial y}\boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega}) \end{bmatrix}, \end{aligned} \tag{2.7}$$

where $\nabla = [\frac{\partial}{\partial x} \quad \frac{\partial}{\partial y}]^\top$ signifies the gradient operator. Putting this assumption in an ℓ^2 error function gives us

$$F_{G,i}(\boldsymbol{\omega}) = \|\nabla \boldsymbol{L}_0(\mathbf{z}) - \nabla \boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega})\|_2^2 \tag{2.8}$$

Combining the two constancy assumptions, we have the following data term

$$\begin{aligned} D_2(\boldsymbol{\omega}) &= \sum_{\boldsymbol{\theta}_i \in \Pi} F_{g,i}(\boldsymbol{\omega}) + \gamma \sum_{\boldsymbol{\theta}_i \in \Pi} F_{G,i}(\boldsymbol{\omega}) \\ &= \sum_{\boldsymbol{\theta}_i \in \Pi} \|\boldsymbol{L}_0(\mathbf{z}) - \boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega})\|_2^2 + \gamma \sum_{\boldsymbol{\theta}_i \in \Pi} \|\nabla \boldsymbol{L}_0(\mathbf{z}) - \nabla \boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega})\|_2^2, \end{aligned} \tag{2.9}$$

which is assembled by summing up the ℓ^2 norms of intensity and gradient differences over all sub-aperture images, i.e., $\boldsymbol{L}_i(\mathbf{z}), \boldsymbol{\theta}_i \in \Pi/\{\boldsymbol{\theta}_0\}$. The parameter γ helps in controlling the contribution of each constancy assumption.

The ℓ^2 norm in Eq. 2.9 can be replaced by the ℓ^1 norm which is more robust against noise and outliers [76]. However, since ℓ^1 is non-smooth leading to the non-existence of the derivative at the point 0, it is more preferable to employ an approximate version of ℓ^1 norm defined as

$$\Psi(s^2) = \sqrt{s^2 + \epsilon},$$

where $\epsilon > 0$ serves as a small regularization parameter, which also allows the derivative of Ψ to be available when $s = 0$. This sub-quadratic function was used in the literature [74,75] showing better performance as compared to ℓ^2 norm and more effective in the term of implementation as compared to ℓ^1. Employing this sub-quadratic penalizer function, Eq. 2.9 is then rewritten as,

$$D_1(\boldsymbol{\omega}) = \sum_{\boldsymbol{\theta}_i \in \Pi} \Psi\big(\|\boldsymbol{L}_0(\mathbf{z}) - \boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega})\|_2^2 \big) + \gamma \sum_{\boldsymbol{\theta}_i \in \Pi} \Psi\big(\|\nabla \boldsymbol{L}_0(\mathbf{z}) - \nabla \boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega})\|_2^2 \big). \tag{2.10}$$

This data term penalizes the constancy assumption by applying the robustification on each SAI pixel and across all available view angles. Instead of applying the robustification perspective-wise, we could combine constancy assumptions of all perspectives in a single robustification,

$$D_1(\boldsymbol{\omega}) = \Psi\Big(\sum_{\boldsymbol{\theta}_i \in \Pi} \|\boldsymbol{L}_0(\mathbf{z}) - \boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega})\|_2^2 \Big) + \gamma \Psi\Big(\sum_{\boldsymbol{\theta}_i \in \Pi} \|\nabla \boldsymbol{L}_0(\mathbf{z}) - \nabla \boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega})\|_2^2 \Big). \tag{2.11}$$

We noticed that this model requires less computational resources as compared to perspective-wise robustification while achieving a good estimation accuracy. The separate robustifications for the two constancy assumptions applied in this model is followed the suggestion from Bruhn and Weickert [77]. This is favored in the situation that the correctness of the two assumptions may not hold true at the same time.

2.2.2 Regularization Term

One of the well-known choices for the smoothness term is Tikhonov prior taking the following form,

$$S(\boldsymbol{\omega}) = \|\Gamma\boldsymbol{\omega}\|_2^2, \tag{2.12}$$

where Γ denotes a regularization operator which is normally chosen as an approximation of i^{th} order derivative. For example, setting Γ to the first order derivative gives us the regularization term $S(\boldsymbol{\omega}) = \|\nabla\boldsymbol{\omega}\|_2^2$. The application of ℓ^2 norm would simplify numerical computation, however, in exchange for the computational advantage, the Tikhonov regularization generally suffers from an over-smooth effect. For the disparity estimation problem, it is preferable to employ ℓ^1 norm, also known as total variation (TV), which can better handle outlier and produces a piece-wise smoothness in the disparity field [74, 78]. Following the similar strategy as for the data fidelity term, the robustification function is employed for alleviating the non-differentiable issue of ℓ^1 norm,

$$S(\boldsymbol{\omega}) = \alpha\ \Psi_{\epsilon_3}(\|\nabla\boldsymbol{\omega}\|_2^2), \tag{2.13}$$

where the parameter α in (2.13) allows controlling the smoothness of the solution.

2.3 Optimization Method

This section discusses the minimization of the variational optimization problems in the previous section. Since both data term and smoothness are differentiable, Euler-Lagrange equations could be applied for minimizing the energy functions [74, 77]. The ultimate goal is transforming the optimization problem into a linear system of equations so that we can effectively solve them with common numerical methods. We start with the simpler version of the problem, i.e., ℓ^2 data-term and Tikhonov regularization with the first-order derivative operator. Through solving this problem, a computation basis including the notation and important transformation are established. This will support the development in the following sections covering the optimization of robust norms, coarse-to-fine strategy, and numerical computation.

2.3.1 Solving Quadratic Data-term and Smoothness Term

Combining the square norm data-term, Eq. 2.9, and the Tikhonov smoothness term, Eq. 2.12, give us the following cost function defined as a functional of the disparity map $\boldsymbol{\omega}(\mathbf{z})$,

$$\begin{aligned} E[\boldsymbol{\omega}] &= \int_{\Omega} \mathcal{L}(\mathbf{z}, \boldsymbol{\omega}, \nabla\boldsymbol{\omega}) d\mathbf{z} \\ &= \int_{\Omega} \sum_{\boldsymbol{\theta}_i \in \Pi} F_{i,g} + \gamma \sum_{\boldsymbol{\theta}_i \in \Pi} F_{i,G} + \alpha \|\nabla\boldsymbol{\omega}\|_2^2 \, d\mathbf{z} \\ &= \int_{\Omega} \sum_{\boldsymbol{\theta}_i \in \Pi} \|\boldsymbol{L}_0(\mathbf{z}) - \boldsymbol{L}_i(\mathbf{z} + \boldsymbol{\theta}_i\boldsymbol{\omega})\|_2^2 + \gamma \sum_{\boldsymbol{\theta}_i \in \Pi} \|\nabla\boldsymbol{L}_0(\mathbf{z}) - \nabla\boldsymbol{L}_i(\mathbf{z} + \boldsymbol{\theta}_i\boldsymbol{\omega})\|_2^2 \\ &\qquad + \alpha \|\nabla\boldsymbol{\omega}\|_2^2 \, d\mathbf{z}, \end{aligned} \tag{2.14}$$

where $\mathcal{L}$ signifies Lagrangian function, and $F_{g,i}$, $F_{G,i}$ are square error functions with regard to the brightness and gradient constancy assumptions respectively. To minimize this energy function, we first approximate the nonlinear functional term $\boldsymbol{L}_i(\mathbf{z} + \boldsymbol{\omega}\boldsymbol{\theta}_i)$ by employing the Taylor series theorem,

$$\begin{aligned} \boldsymbol{L}_i(\mathbf{z} + \boldsymbol{\omega}\boldsymbol{\theta}_i) &\approx \boldsymbol{L}_i(\mathbf{z}) + \rho_i\boldsymbol{\omega}\frac{\partial}{\partial x}\boldsymbol{L}_i(\mathbf{z}) + \tau_i\boldsymbol{\omega}\frac{\partial}{\partial y}\boldsymbol{L}_i(\mathbf{z}) \\ &= \boldsymbol{L}_i(\mathbf{z}) + \boldsymbol{\omega}\boldsymbol{\theta}_i^{\top}\nabla\boldsymbol{L}_i(\mathbf{z}) \end{aligned} \tag{2.15}$$

Let us define $\mathbf{w} = [\boldsymbol{\omega} \quad 1]^{\top}$ and introduce the term

$$\nabla_{\boldsymbol{\theta}_i}\boldsymbol{L} = \begin{bmatrix} \boldsymbol{\theta}_i^{\top}\nabla\boldsymbol{L}_i(\mathbf{z}) \\ \boldsymbol{L}_i(\mathbf{z}) - \boldsymbol{L}_0(\mathbf{z}) \end{bmatrix}. \tag{2.16}$$

We rewrite the photo constancy error function as,

$$\begin{aligned} F_{g,i}(\boldsymbol{\omega}) &= \|\boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\omega}\boldsymbol{\theta}_i) - \boldsymbol{L}_0(\mathbf{z})\|_2^2 \\ &\approx \left\|\boldsymbol{L}_i(\mathbf{z}) - \boldsymbol{L}_0(\mathbf{z}) + \boldsymbol{\omega}\boldsymbol{\theta}_i^T\nabla\boldsymbol{L}_i(\mathbf{z})\right\|_2^2 \\ &= \left\|\mathbf{w}^T\nabla_{\theta_i}\boldsymbol{L}\right\|_2^2 \\ &= \mathbf{w}^\intercal\left(\nabla_{\theta_i}\boldsymbol{L}\nabla_{\theta_i}^\intercal\boldsymbol{L}\right)\mathbf{w} \\ &= \mathbf{w}^\intercal\mathbf{J}_{g,i}\mathbf{w}, \end{aligned}$$

where $\mathbf{J}_{g,i} = \nabla_{\theta_i}\boldsymbol{L}\nabla_{\theta_i}^\intercal\boldsymbol{L}$ denotes a Light field motion tensor computed for the photo constancy assumption. This well-known motion tensor notation [75, 79] allows us to simplify the expression in combination of perspective constancy constraints. $\mathbf{J}_{g,i}$ is a 2×2 symmetric matrix defined for each pixel position $\mathbf{z}\in\Omega$ and for specific view point $\boldsymbol{\theta}_i\in\Pi$. For the gradient constancy assumption, we follow a similar linearization process applying the first-order Tailor expansion,

$$\begin{aligned} \nabla\boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega}) &= \begin{bmatrix} \frac{\partial}{\partial x}\boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega}) \\ \frac{\partial}{\partial y}\boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega}) \end{bmatrix} \\ &\approx \begin{bmatrix} \frac{\partial}{\partial x}\boldsymbol{L}_i(\mathbf{z}) + \boldsymbol{\omega}\boldsymbol{\theta}_i^T\nabla\frac{\partial}{\partial x}\boldsymbol{L}_i(\mathbf{z}) \\ \frac{\partial}{\partial y}\boldsymbol{L}_i(\mathbf{z}) + \boldsymbol{\omega}\boldsymbol{\theta}_i^T\nabla\frac{\partial}{\partial y}\boldsymbol{L}_i(\mathbf{z}) \end{bmatrix}, \end{aligned} \tag{2.17}$$

The gradient constancy error function is rewritten as,

$$\begin{aligned} F_{G,i} &= \|\nabla\boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega}) - \nabla\boldsymbol{L}_0(\mathbf{z})\|_2^2 \\ &\approx \left(\frac{\partial}{\partial x}\boldsymbol{L}_i(\mathbf{z}) - \frac{\partial}{\partial x}\boldsymbol{L}_0(\mathbf{z}) + \boldsymbol{\omega}\boldsymbol{\theta}_i^T\nabla\frac{\partial}{\partial x}\boldsymbol{L}_i(\mathbf{z})\right)^2 + \left(\frac{\partial}{\partial y}\boldsymbol{L}_i(\mathbf{z}) - \frac{\partial}{\partial y}\boldsymbol{L}_o(\mathbf{z}) + \boldsymbol{\omega}\boldsymbol{\theta}_i^T\nabla\frac{\partial}{\partial y}\boldsymbol{L}_i(\mathbf{z})\right)^2 \\ &= \mathbf{w}^T\nabla_{\theta_i}\boldsymbol{L}_x\nabla_{\theta_i}^\intercal\boldsymbol{L}_x\mathbf{w} + \mathbf{w}^T\nabla_{\theta_i}\boldsymbol{L}_y\nabla_{\theta_i}^\intercal\boldsymbol{L}_y\mathbf{w} \\ &= \mathbf{w}^\intercal\left(\nabla_{\theta_i}\boldsymbol{L}_x\nabla_{\theta_i}^\intercal\boldsymbol{L}_x + \nabla_{\theta_i}\boldsymbol{L}_y\nabla_{\theta_i}^\intercal\boldsymbol{L}_y\right)\mathbf{w} \\ &= \mathbf{w}^\intercal\mathbf{J}_{G,i}\mathbf{w}, \end{aligned}$$

with $\mathbf{J}_{G,i}$ is a Light field motion tensor computed with regard to the gradient constancy assumption at a viewpoint $\boldsymbol{\theta}_i\in\Pi$. The terms $\nabla_{\theta_i}\boldsymbol{L}_x$ and $\nabla_{\theta_i}\boldsymbol{L}_y$ are computed as follow

$$\nabla_{\theta_i}\boldsymbol{L}_\eta = \begin{bmatrix} \boldsymbol{\theta}_i^T\nabla\frac{\partial}{\partial\eta}\boldsymbol{L}_i(\mathbf{z}) \\ \frac{\partial}{\partial\eta}\boldsymbol{L}_i(\mathbf{z}) - \frac{\partial}{\partial\eta}\boldsymbol{L}_0(\mathbf{z}) \end{bmatrix}, \quad \eta\in\{x,y\},$$

which is similar to Eq. 2.16, except that the directional gradient operator (∇_{θ_i}) is applied to spatial derivative versions of 4D LF, i.e., $\boldsymbol{L}_x = \frac{\partial}{\partial x}\boldsymbol{L}(\mathbf{z},\boldsymbol{\theta})$ and $\boldsymbol{L}_y = \frac{\partial}{\partial y}\boldsymbol{L}(\mathbf{z},\boldsymbol{\theta})$, instead of the 4D LF itself. Let us define the total Light field motion tensors **J** as the sum of the tensors at all viewpoints,

$$\mathbf{J} = \mathbf{J}_g + \gamma \mathbf{J}_G, \quad \mathbf{J}_g = \sum_{\theta_i \in \Pi} \mathbf{J}_{g,i}, \quad \mathbf{J}_G = \sum_{\theta_i \in \Pi} \mathbf{J}_{G,i}. \tag{2.18}$$

We then approximate the Lagrangian $\mathcal{L}$ in Eq.2.14 as

$$\begin{aligned} \mathcal{L}(\mathbf{z}, \omega, \nabla\omega) &\approx \mathbf{w}^\intercal \mathbf{J}_g \mathbf{w} + \gamma \mathbf{w}^\intercal \mathbf{J}_G \mathbf{w} + \alpha \left\|\nabla\omega\right\|_2^2 \\ &= \mathbf{w}^\intercal \mathbf{J} \mathbf{w} + \alpha \left\|\nabla\omega\right\|_2^2, \end{aligned} \tag{2.19}$$

The solution ω is the stationary point of the energy functional Eq. 2.14, the following Euler-Lagrange equation must be satisfied,

$$0 = \frac{\partial}{\partial\omega}\mathcal{L} - \frac{\partial}{\partial x}\frac{\partial}{\partial\omega_x}\mathcal{L} - \frac{\partial}{\partial y}\frac{\partial}{\partial\omega_y}\mathcal{L}, \tag{2.20}$$

with ω_x and ω_y denote the x and y derivative of the disparity map $\omega(\mathbf{z})$. For the computation of the partial derivative, we first expand the Eq. 2.19,

$$\mathcal{L}(\mathbf{z}, \omega, \nabla\omega) = \mathbf{J}_{11}\omega^2 + 2\mathbf{J}_{12}\omega + \mathbf{J}_{22} + \alpha(\omega_x^2 + \omega_y^2),$$

where $\mathbf{J}_{11}$, $\mathbf{J}_{12}$, and $\mathbf{J}_{22}$ are extracted from the 2×2 symmetric LF motion tensor,

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{11} & \mathbf{J}_{12} \\ \mathbf{J}_{21} & \mathbf{J}_{22} \end{bmatrix}.$$

The partial derivatives in Eq. 2.20 are computed as follows,

$$\begin{aligned} \frac{\partial}{\partial\omega}\mathcal{L} &= 2\mathbf{J}_{11}\omega + 2\mathbf{J}_{12}, \\ \frac{\partial}{\partial x}\frac{\partial}{\partial\omega_x}\mathcal{L} + \frac{\partial}{\partial y}\frac{\partial}{\partial\omega_y}\mathcal{L} &= 2\alpha\frac{\partial}{\partial x}\omega_x + 2\alpha\frac{\partial}{\partial y}\omega_y, \end{aligned} \tag{2.21}$$

which turns Eq. 2.20 to,

$$0 = \mathbf{J}_{11}\omega + \mathbf{J}_{12} - \alpha \mathbf{div}\,(\nabla\omega). \tag{2.22}$$

Since this equation is applied to each pixel in the reference SAI and the divergence term $\mathbf{div}\,(\nabla\omega)$ can be effectively realized by linear operators, a linear system of equations would be formed as will be discussed in Sec. 2.4.

2.3.2 Solving Sub-quadratic Data Term and Smoothness Term

In the following, the optimization of robustification data-term and regularization term is discussed. It is also our final choice of the energy function which provides the most accurate estimation of the disparity map. Let us start with the general form of the energy function:

$$\begin{aligned} E[\boldsymbol{\omega}] &= \int_{\Omega} \mathcal{L}(\mathbf{z}, \boldsymbol{\omega}, \nabla\boldsymbol{\omega}) d\mathbf{z} \\ &= \int_{\Omega} \Psi\left(\sum_{\theta_i \in \Pi} F_{i,g}\right) + \gamma\Psi\left(\sum_{\theta_i \in \Pi} F_{i,G}\right) + \alpha\Psi\left(\|\nabla\boldsymbol{\omega}\|_2^2\right) d\mathbf{z} \end{aligned} \tag{2.23}$$

Follow the linearization strategy and notation from the previous section, we rewrite the Lagrangian function as follows

$$\mathcal{L}(\mathbf{z}, \boldsymbol{\omega}, \nabla\boldsymbol{\omega}) = \Psi\big(\mathbf{w}^\top \mathbf{J}_g \mathbf{w}\big) + \gamma\Psi\big(\mathbf{w}^\top \mathbf{J}_G \mathbf{w})\big) + \alpha\Psi\big(\, \|\nabla\boldsymbol{\omega}\|_2^2\,\big), \tag{2.24}$$

The partial derivatives in Eq. 2.20 are computed as follows

$$\begin{aligned} \frac{\partial}{\partial\boldsymbol{\omega}}\mathcal{L} &= 2\Psi'(\mathbf{w}^\top \mathbf{J}_g \mathbf{w})\left(\mathbf{J}_{g,11}\boldsymbol{\omega} + \mathbf{J}_{g,12}\right) + 2\gamma\Psi'(\mathbf{w}^\top \mathbf{J}_G \mathbf{w})\left(\mathbf{J}_{G,11}\boldsymbol{\omega} + \mathbf{J}_{G,12}\right), \\ \frac{\partial}{\partial x}\frac{\partial}{\partial\boldsymbol{\omega}_x}\mathcal{L} + \frac{\partial}{\partial y}\frac{\partial}{\partial\boldsymbol{\omega}_y}\mathcal{L} &= 2\alpha\frac{\partial}{\partial x}\left(\Psi'(\,\|\nabla\boldsymbol{\omega}\|_2^2\,)\boldsymbol{\omega}_x\right) + 2\alpha\frac{\partial}{\partial y}\left(\Psi'(\,\|\nabla\boldsymbol{\omega}\|_2^2\,)\boldsymbol{\omega}_y\right), \end{aligned} \tag{2.25}$$

where $\mathbf{J}_{*,11}, \mathbf{J}_{*,12}$ denotes the components of the LF motion tensor $\mathbf{J}_*$ and $\Psi'(s^2)$ is the derivative of the sub-quadratic function, $\Psi'(s^2) = \frac{1}{2\sqrt{s^2+\epsilon}} = \frac{1}{2\Psi(s^2)}$. For simplifying the notation let us assign,

$$\Psi'_g = \Psi'\big(\mathbf{w}^\top \mathbf{J}_g \mathbf{w}\big), \quad \Psi'_G = \Psi'\big(\mathbf{w}^\top \mathbf{J}_G \mathbf{w}\big), \quad \Psi'_s = \Psi'\big(\nabla^\top\boldsymbol{\omega}\nabla\boldsymbol{\omega}\big), \tag{2.26}$$

The Euler-Lagrange equation for the functional in Eq. 2.23 reads,

$$\begin{aligned} 0 &= \frac{\partial}{\partial\boldsymbol{\omega}}\mathcal{L} - \frac{\partial}{\partial x}\frac{\partial}{\partial\boldsymbol{\omega}_x}\mathcal{L} - \frac{\partial}{\partial y}\frac{\partial}{\partial\boldsymbol{\omega}_y}\mathcal{L} \\ &= \Psi'_g \cdot (\boldsymbol{\omega}\mathbf{J}_{g,11} + \mathbf{J}_{g,12}) + \gamma\Psi'_G \cdot (\boldsymbol{\omega}\mathbf{J}_{G,11} + \mathbf{J}_{G,12}) - \alpha\,\mathbf{div}\,(\Psi'_s\nabla\boldsymbol{\omega}), \end{aligned} \tag{2.27}$$

with the sign $\cdot$ is introduced to avoid an ambiguity in the representation of Ψ'_* as a function. It is obvious that Eq. 2.27 is non-linear with regard to $\boldsymbol{\omega}$ due to the non-linear function Ψ'_*.

This non-linearity disables the construction of system of linear equations for optimizing the cost function in Eq. 2.23. To overcome this issue, we follow the previous works [74, 80] to apply a fixed point iteration strategy,

$$0 = \Psi_g'^{(k)}\left(\boldsymbol{\omega}^{(k+1)}\mathbf{J}_{g,11} + \mathbf{J}_{g,12}\right) + \gamma\Psi_G'^{(k)}\left(\boldsymbol{\omega}^{(k+1)}\mathbf{J}_{G,11} + \mathbf{J}_{G,12}\right) - \alpha\,\mathbf{div}\left(\Psi_s'^{(k)}\nabla\boldsymbol{\omega}^{(k+1)}\right), \quad (2.28)$$

which computes the values of Ψ_*' using the solution from the previous iteration, i.e., $\boldsymbol{\omega}^{(k)}$. The non-linearity term $\Psi_*'^{(k)}$ is then treated as a constant in the $k+1^{th}$ iteration searching for the unknown $\boldsymbol{\omega}^{(k+1)}$. Assembling Eq. 2.28 for all pixels in the reference SAI allows us to form a sparse system of linear equations which can be solved effectively, as discussed in Section 2.4.

2.3.3 Coarse-to-fine Warping Strategy

In the previous section, we shown that applying Tailor expansion technique helps to turn the functional $\boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega})$ into the linear combination of the mapping function $\boldsymbol{\omega}$. Although this linearization strategy simplifies the optimization problem, it takes the risk of loosing accuracy due to the approximation of the functional. To overcome this issue, a coarse-to-fine warping strategy [74, 75] could be applied, in which the approximation of the functional is postponed to the numerical step and disparity map is computed and gradually refined from the coarsest scale to the finest scale.

Let us start with the Lagrangian without linearization,

$$\begin{aligned}\mathcal{L}(\mathbf{z},\boldsymbol{\omega},\nabla\boldsymbol{\omega}) =& \Psi\Big(\sum_{\Pi}\|\boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega}) - \boldsymbol{L}_0(\mathbf{z})\|_2^2\Big) + \gamma\Psi\Big(\sum_{\Pi}\|\nabla\boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega}) - \nabla\boldsymbol{L}_0(\mathbf{z})\|_2^2\Big)\\ &+ \alpha\Psi\Big(\|\nabla\boldsymbol{\omega}\|_2^2\Big).\end{aligned}$$

Here, let us rewrite $\widehat{\boldsymbol{L}}_i = \boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega}) = \boldsymbol{L}(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega},\boldsymbol{\theta}_i)$ to simplify the notation. It follows that $\widehat{\boldsymbol{L}}_0 = \boldsymbol{L}_0(\mathbf{z}) = \boldsymbol{L}(\mathbf{z},\boldsymbol{\theta}_0)$. The partial derivative $\frac{\partial}{\partial\boldsymbol{\omega}}\mathcal{L}$ is computed as,

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{\omega}}\mathcal{L} =& 2\Psi'\left(\sum_{\Pi}\left\|\widehat{\boldsymbol{L}}_i - \boldsymbol{L}_0\right\|_2^2\right)\sum_{\boldsymbol{\theta}_i\in\Pi}\left(\widehat{\boldsymbol{L}}_i - \widehat{\boldsymbol{L}}_0\right)\frac{d}{d\boldsymbol{\omega}}\widehat{\boldsymbol{L}}_i\\ &+ 2\gamma\Psi'\left(\sum_{\Pi}\left\|\nabla\widehat{\boldsymbol{L}}_i - \nabla\widehat{\boldsymbol{L}}_0\right\|_2^2\right)\sum_{\Pi}\left(\nabla\widehat{\boldsymbol{L}}_i - \nabla\widehat{\boldsymbol{L}}_0\right)^{\top}\frac{d}{d\boldsymbol{\omega}}\nabla\widehat{\boldsymbol{L}}_i,\end{aligned} \quad (2.29)$$

while the other partial derivatives, i.e., $\frac{\partial}{\partial x}\frac{\partial}{\partial\omega_x}\mathcal{L}$, $\frac{\partial}{\partial y}\frac{\partial}{\partial\omega_y}\mathcal{L}$, are the same as in Eq. 2.25. The

Euler-Lagrange equation reads

$$0 = \tilde{\Psi}'_g \cdot \sum_{\Pi} (\widehat{L}_i - \widehat{L}_0) \frac{d}{d\boldsymbol{\omega}} \widehat{L}_i + \gamma \tilde{\Psi}'_G \cdot \sum_{\Pi} (\nabla \widehat{L}_i - \nabla \widehat{L}_0)^{\mathsf{T}} \frac{d}{d\boldsymbol{\omega}} \nabla \widehat{L}_i - \alpha \mathbf{div}\left(\Psi'_s \nabla \boldsymbol{\omega}\right), \quad (2.30)$$

where $\tilde{\Psi}'_g = \Psi'\left(\sum_{\Pi} \left\| \widehat{L}_i - \widehat{L}_0 \right\|_2^2\right)$ and $\tilde{\Psi}'_G = \Psi'\left(\sum_{\Pi} \left\| \nabla \widehat{L}_i - \nabla \widehat{L}_0 \right\|_2^2\right)$.

Regarding the derivative of $\widehat{L}_i$ by $\boldsymbol{\omega}$, by definition

$$\frac{d}{d\boldsymbol{\omega}} \widehat{L}_i = \frac{d}{d\boldsymbol{\omega}} \boldsymbol{L}_i(\mathbf{z} + \boldsymbol{\omega}\boldsymbol{\theta}_i) = \lim_{\Delta\omega \to 0} \frac{\boldsymbol{L}_i(\mathbf{z} + \boldsymbol{\omega}\boldsymbol{\theta}_i + \Delta\boldsymbol{\omega}\boldsymbol{\theta}_i) - \boldsymbol{L}_i(\mathbf{z} + \boldsymbol{\omega}\boldsymbol{\theta}_i)}{\Delta\boldsymbol{\omega}},$$

which is actually the directional derivative of $\widehat{L}_i(\mathbf{z})$ at the point $\mathbf{z} + \boldsymbol{\omega}\boldsymbol{\theta}_i$ in the direction of $\boldsymbol{\theta}_i$. Therefore, it reads

$$\frac{d}{d\boldsymbol{\omega}} \widehat{L}_i = \boldsymbol{\theta}_i^{\mathsf{T}} \nabla \widehat{L}_i.$$

In a similar manner, the derivative of $\nabla \widehat{L}_i$ by $\boldsymbol{\omega}$ reads

$$\frac{d}{d\boldsymbol{\omega}} \nabla \widehat{L}_i = \mathbf{H}\, \widehat{L}_i \boldsymbol{\theta}_i,$$

where $\mathbf{H}$ is the 2×2 Hessian matrix, $\mathbf{H} = \begin{bmatrix} \frac{\partial}{\partial x}\frac{\partial}{\partial x} & \frac{\partial}{\partial x}\frac{\partial}{\partial y} \\ \frac{\partial}{\partial y}\frac{\partial}{\partial x} & \frac{\partial}{\partial y}\frac{\partial}{\partial y} \end{bmatrix}$. The Eq. 2.30 becomes

$$0 = \Psi'_g \sum_{\boldsymbol{\theta}_i \in \Pi} (\widehat{L}_i - \widehat{L}_0) \boldsymbol{\theta}_i^T \nabla \widehat{L}_i + \gamma \Psi'_G \sum_{\boldsymbol{\theta}_i \in \Pi} (\nabla \widehat{L}_i - \nabla \widehat{L}_0)^{\mathsf{T}} \mathbf{H}\, \widehat{L}_i \boldsymbol{\theta}_i - \alpha \mathbf{div}\left(\Psi'_i \nabla \boldsymbol{\omega}\right). \quad (2.31)$$

This Euler-Lagrange equation is much more complex than the one with linearization (Eq. 2.27). The equation is non-linear with regard to $\boldsymbol{\omega}$ due to the robust norm function Ψ and the warping SAI $\widehat{L}_i = \boldsymbol{L}(\mathbf{z} + \boldsymbol{\theta}_i \boldsymbol{\omega}, \boldsymbol{\theta}_i)$. To construct a system of linear equations, we follows the approach of Brox et al. [74] which makes use of an incremental computation based on a multi-scale fixed-point iteration. As suggested in [75], this incremental computation can be broken down into three steps. First, we apply a fix-pointed iteration on $\boldsymbol{\omega}$

$$\begin{aligned} 0 = \; & \Psi_g'^{(t+1)} \sum_{\boldsymbol{\theta}_i \in \Pi} (\widehat{L}_i^{(t+1)} - \widehat{L}_0) \boldsymbol{\theta}_i^T \nabla \widehat{L}_i^{(t)} + \gamma \Psi_G'^{(t+1)} \sum_{\boldsymbol{\theta}_i \in \Pi} (\nabla \widehat{L}_i^{(t+1)} - \nabla \widehat{L}_0)^{\mathsf{T}} \mathbf{H}\, \widehat{L}_i^{(t)} \boldsymbol{\theta}_i \\ & - \alpha \mathbf{div}\left(\Psi_i'^{(t+1)} \nabla \boldsymbol{\omega}^{(t+1)}\right), \end{aligned} \quad (2.32)$$

which employs the estimated disparity map from the previous iteration, i.e., $\boldsymbol{\omega}^{(t)}$, for the computation of the gradient $\nabla \widehat{L}^{(t)}$ and hessian $\mathbf{H}\,\widehat{L}^{(t)}$, which are associated to the photometric and the gradient assumptions respectively.

In the second step, let us introduce an increment $\boldsymbol{\delta\omega}^{(k)}$ which accounts for the change of $\boldsymbol{\omega}$ between step $k^{(th)}$ and $k+1^{(th)}$, such as $\boldsymbol{\omega}^{(k+1)} = \boldsymbol{\omega}^{(k)} + \boldsymbol{\delta\omega}^{(k)}$. This allows us to transform the unknown disparity amount $\boldsymbol{\omega}^{(k+1)}$ into the known disparity $\boldsymbol{\omega}^{(k)}$ and an unknown incremental amount $\boldsymbol{\delta\omega}^{(k)}$. Assuming that $\boldsymbol{\delta\omega}$ only accounts for a small displacement, the linearization strategy can be applied

$$\begin{aligned} \widehat{L}_i^{(t+1)} &= \boldsymbol{L}_i(\mathbf{z} + \boldsymbol{\theta}_i\boldsymbol{\omega}^{(t+1)}) \\ &= \boldsymbol{L}_i(\mathbf{z} + \boldsymbol{\theta}_i\boldsymbol{\omega}^{(t)} + \boldsymbol{\theta}_i\boldsymbol{\delta\omega}^{(t)}) \\ &\approx \boldsymbol{L}_i(\mathbf{z} + \boldsymbol{\theta}_i\boldsymbol{\omega}^{(t)}) + \boldsymbol{\delta\omega}^{(t)}\boldsymbol{\theta}_i^{\top}\nabla\boldsymbol{L}_i(\mathbf{z} + \boldsymbol{\theta}_i\boldsymbol{\omega}^{(t)}) \\ &= \widehat{L}_0 + \begin{bmatrix}\boldsymbol{\delta\omega}^{(t)} & 1\end{bmatrix}\nabla_{\theta_i}\widehat{L}^{(t)}, \end{aligned} \tag{2.33}$$

with LF gradient operator ∇_{θ_i} follows Eq. 2.16. The gradient of the warped LF can also be approximated in a similar manner

$$\begin{aligned} \nabla\widehat{L}_i^{(t+1)} &= \nabla\boldsymbol{L}_i(\mathbf{z} + \boldsymbol{\theta}_i\boldsymbol{\omega}^{(t+1)}) \\ &\approx \nabla\widehat{L}_0 + \begin{bmatrix}\nabla_{\theta_i}^{\top}\widehat{L}_x \\ \nabla_{\theta_i}^{\top}\widehat{L}_y\end{bmatrix}\begin{bmatrix}\boldsymbol{\delta\omega}^{(t)} \\ 1\end{bmatrix}, \end{aligned} \tag{2.34}$$

where $\nabla_{\theta_i}^{\top}\widehat{L}_*$ denotes the transpose version of LF gradient operator applied to the derivatives $\widehat{L}_\eta = \frac{\partial}{\partial\eta}\widehat{L}(\mathbf{z},\boldsymbol{\theta}), \quad \eta \in \{x, y\}$. We then repeat the definition of the total LF motion tensor for photometric and gradient constancy assumptions applied to the warped LF

$$\widehat{\mathbf{J}} = \widehat{\mathbf{J}}_g + \gamma\widehat{\mathbf{J}}_G, \quad \widehat{\mathbf{J}}_g = \sum_{\theta_i\in\Pi}\nabla_{\theta_i}^{T}\widehat{L}\nabla_{\theta_i}\widehat{L}, \quad \widehat{\mathbf{J}}_G = \sum_{\theta_i\in\Pi}\nabla_{\theta_i}^{T}\widehat{L}_x\nabla_{\theta_i}\widehat{L}_x + \nabla_{\theta_i}^{T}\widehat{L}_y\nabla_{\theta_i}\widehat{L}_y \tag{2.35}$$

The summation in Eq. 2.32 can be rewritten by employing the total LF motion tensor notation

$$\begin{aligned} \sum_{\theta_i\in\Pi}\left(\widehat{L}_i^{(t+1)} - \widehat{L}_0\right)\boldsymbol{\theta}_i^{\top}\nabla\widehat{L}_i^{(t)} &\approx \sum_{\theta_i\in\Pi}\begin{bmatrix}\boldsymbol{\delta\omega}^{(t)} & 1\end{bmatrix}\begin{bmatrix}\boldsymbol{\theta}_i^{\top}\nabla\widehat{L}_i(\mathbf{z}) \\ \widehat{L}_i(\mathbf{z}) - \widehat{L}_0(\mathbf{z})\end{bmatrix}\boldsymbol{\theta}_i^{\top}\nabla\widehat{L}_i^{(t)} \\ &= \widehat{\mathbf{J}}_{g,11}^{(t)}\boldsymbol{\delta\omega}^{(t)} + \widehat{\mathbf{J}}_{g,12}^{(t)} \quad , \end{aligned} \tag{2.36}$$

and

$$\sum_{\theta_i \in \Pi} \left(\nabla \widehat{L}_i^{(t+1)} - \nabla \widehat{L}_0\right)^\top \mathbf{H}\, \widehat{L}_i^{(t)} \theta_i \approx \quad \widehat{\mathbf{J}}_{G,11}^{(t)} \boldsymbol{\delta\omega}^{(t)} + \widehat{\mathbf{J}}_{G,12}^{(t)}. \tag{2.37}$$

The derivatives of robustification function Ψ_* are computed as

$$\begin{aligned} \Psi_g'^{(t+1)} &\approx \quad \widehat{\Psi}_g'^{(t)} = \Psi'\left(\left[\boldsymbol{\delta\omega}^{(t)} \quad 1\right] \widehat{\mathbf{J}}_g^{(t)} \left[\boldsymbol{\delta\omega}^{(t)} \quad 1\right]^\top\right), \\ \Psi_G'^{(t+1)} &\approx \quad \widehat{\Psi}_G'^{(t)} = \Psi'\left(\left[\boldsymbol{\delta\omega}^{(t)} \quad 1\right] \widehat{\mathbf{J}}_G^{(t)} \left[\boldsymbol{\delta\omega}^{(t)} \quad 1\right]^\top\right), \\ \Psi_s'^{(t+1)} &\approx \quad \widehat{\Psi}_s'^{(t)} = \Psi'\left(\left\|\nabla \boldsymbol{\omega}^{(k)} + \nabla \boldsymbol{\delta\omega}^{(k)}\right\|_2^2\right). \end{aligned} \tag{2.38}$$

We can then rewrite Eq. 2.32 into a much cleaner form

$$0 = \widehat{\Psi}_g'^{(t)} \cdot \left(\widehat{\mathbf{J}}_{g,11}^{(t)} \boldsymbol{\delta\omega}^{(t)} + \widehat{\mathbf{J}}_{g,12}^{(t)}\right) + \gamma \widehat{\Psi}_G'^{(t)} \cdot \left(\widehat{\mathbf{J}}_{G,11}^{(t)} \boldsymbol{\delta\omega}^{(t)} + \widehat{\mathbf{J}}_{G,12}^{(t)}\right) - \alpha \mathbf{div}\left(\widehat{\Psi}_s'^{(t)} \cdot (\nabla \boldsymbol{\omega}^{(t)} + \nabla \boldsymbol{\delta\omega}^{(t)})\right). \tag{2.39}$$

Important changes of Eq. 2.39 as compared to Eq. 2.32 are the switch from the unknown disparity value $\boldsymbol{\omega}^{(t+1)}$ to the unknown incremental amount $\boldsymbol{\delta\omega}^t$ and the removal of non-linearity in the warped LF (i.e., $\widehat{L}_i^{(t+1)}$, and $\nabla \widehat{L}_i^{(t+1)}$). However the above equation is still non-linear with regard to $\boldsymbol{\delta\omega}^{(t)}$ due to the nonlinear function $\widehat{\Psi}_*'^{(t)}$. To overcome this issue, a fixed-point iteration on an inner loop searching for $\boldsymbol{\delta\omega}^{(t)}$ is applied

$$\begin{aligned} 0 = \quad & \widehat{\Psi}_g'^{(t,l)} \cdot \left(\widehat{\mathbf{J}}_{g,11}^{(t)} \boldsymbol{\delta\omega}^{(t,l+1)} + \widehat{\mathbf{J}}_{g,12}^{(t)}\right) + \gamma \widehat{\Psi}_G'^{(t)} \cdot \left(\widehat{\mathbf{J}}_{G,11}^{(t)} \boldsymbol{\delta\omega}^{(t,l+1)} + \widehat{\mathbf{J}}_{G,12}^{(t)}\right) \\ & - \alpha \mathbf{div}\left(\widehat{\Psi}_s'^{(t,l)} \cdot (\nabla \boldsymbol{\omega}^{(t)} + \nabla \boldsymbol{\delta\omega}^{(t,l+1)})\right), \end{aligned} \tag{2.40}$$

where the superscript (t, l) denotes the outer loop $t^{(th)}$ searching for $\boldsymbol{\omega}$ and then inner loop l^{th} searching for $\boldsymbol{\delta\omega}$. The derivatives $\hat{\Psi}_*'$ are computed using the known value of $\boldsymbol{\omega}^{(t)}$ and $\boldsymbol{\delta\omega}^{(l)}$ estimated from the previous iteration. Taking a closer look, one can see the similarity between the system of equation in Eq. 2.28 and Eq. 2.40. This similarity suggests that we can reuse the solver of the former for solving the latter taking into account the following key observations. First, the total LF motion tensors in the former equation are computed from the original LF while the latter computes them from the warped LF. Second, the smoothness term includes a so call initial disparity. Both equations can be derived from the following form of the energy function

$$E[\boldsymbol{\omega}] = \int_\Omega \underbrace{\Psi\left(\mathbf{w}^\top \mathbf{J}_g \mathbf{w}\right) + \gamma \Psi\left(\mathbf{w}^\top \mathbf{J}_G \mathbf{w}\right)}_{D(\mathbf{J}_g, \mathbf{J}_G, \boldsymbol{\omega})} + \underbrace{\alpha \Psi\left(\|\nabla \boldsymbol{\omega}_0 + \nabla \boldsymbol{\omega}\|_2^2\right)}_{S(\boldsymbol{\omega}_0, \boldsymbol{\omega})} d\mathbf{z}, \tag{2.41}$$

with the initial disparity map $\boldsymbol{\omega}_0$ helps in the regularization of $\boldsymbol{\omega}$. For Eq. 2.28, $\boldsymbol{\omega}_0$ is set to 0. Here let us define auxiliary data and smoothness functions $D(\mathbf{J}_g, \mathbf{J}_G, \boldsymbol{\omega})$ and $S(\boldsymbol{\omega}_0, \boldsymbol{\omega})$.

For solving the Eq. 2.40, the outer-inner iterations are incorporated, in the third step, into a coarse-to-fine warping scheme [74, 75] applied on a set of spatially scaled LF (referred to as LF spatial pyramid). The outer loop starts with the initialization of the disparity map at each level in the pyramid started from the coarsest level. The inner loop is for the refinement of the initial map by estimating the incremental disparity amount. Algorithm 1 outlines this coarse-to-fine strategy.

Algorithm 1: Coarse-to-fine warping Strategy

```
Input: ${\{\boldsymbol{L}\}_t,\ t = 0, 1..T}$
Output: $\boldsymbol{\omega}$
1  for $t := T$ downto $0$ do
2      if $t = T$ then
3          $\{\boldsymbol{\omega}\}_t := 0$                                  ▷ Initializing with zeros
4      else
5          $\{\boldsymbol{\omega}\}_t :=\uparrow \{\widetilde{\boldsymbol{\omega}}\}_{t+1}$          ▷ Up-sampling $\{\widetilde{\boldsymbol{\omega}}\}_{t+1}$
6      end
7      $\widehat{L}(\mathbf{z}, \boldsymbol{\theta}_i) := \left\{\boldsymbol{L}(\mathbf{z} + \boldsymbol{\theta}_i\{\boldsymbol{\omega}\}_t, \boldsymbol{\theta}_i)\right\}_t$          ▷ Wapring with $\{\boldsymbol{\omega}\}_t$
8      $\widehat{\mathbf{J}}_g := \sum_{\boldsymbol{\theta}_i \in \Pi} \nabla^T_{\boldsymbol{\theta}_i} \widehat{L} \nabla_{\boldsymbol{\theta}_i} \widehat{L}$
9      $\widehat{\mathbf{J}}_G := \sum_{\boldsymbol{\theta}_i \in \Pi} \nabla^T_{\boldsymbol{\theta}_i} \widehat{L}_x \nabla_{\boldsymbol{\theta}_i} \widehat{L}_x + \nabla^T_{\boldsymbol{\theta}_i} \widehat{L}_y \nabla_{\boldsymbol{\theta}_i} \widehat{L}_y$
10     $\left\{\boldsymbol{\delta}\widetilde{\boldsymbol{\omega}}\right\}_t := \arg\min_{\boldsymbol{\delta\omega}} \int_{\Omega} D\Big(\widehat{\mathbf{J}}_g, \widehat{\mathbf{J}}_G, \boldsymbol{\delta\omega}\Big) + S\Big(\{\boldsymbol{\omega}\}_t, \boldsymbol{\delta\omega}\Big)\, d\mathbf{z}$   ▷ Optimizing the Eq. 2.41
11     $\{\widetilde{\boldsymbol{\omega}}\}_t := \{\boldsymbol{\omega}\}_t + \{\boldsymbol{\delta}\widetilde{\boldsymbol{\omega}}\}_t$
12 end
```

A set of low-resolution light field image, denoted as $\{\boldsymbol{L}\}_t,\ t = 0, 1, ..T$, is first generated from the input 4D light field $\boldsymbol{L}$. T denotes the maximum coarse-to-fine level. $\{\boldsymbol{L}\}_t$ has the same angular resolution as of $\boldsymbol{L}$, but has spatial resolution scaled by ζ^t. The parameter ζ $\big(\zeta \in \mathbb{R}, 0 < \zeta < 1\big)$ denotes a predefined scaling factor which remains constant in all warping levels. Instead of calculating the disparity map $\boldsymbol{\omega}$ directly from the light field $\boldsymbol{L}$, a refined disparity map $\{\widetilde{\boldsymbol{\omega}}\}_t$ is estimated for each of low-resolution light field $\{\boldsymbol{L}\}_t$ starting from the coarsest level ($t = T$). Given a light field $\boldsymbol{L}$ with the spatial resolution of $H \times W$, the spatial resolution of a light field $\big(\{\boldsymbol{L}\}_t\big)$ at a warping level t is $\zeta^t H \times \zeta^t W$. This is also the resolution of pre-estimated disparity map $\big(\{\boldsymbol{\omega}\}_t\big)$ and refined disparity map $\big(\{\widetilde{\boldsymbol{\omega}}\}_t\big)$ at the warping level t. The pre-estimated disparity map at the coarsest level $(\{\boldsymbol{\omega}\}_T)$ is initialized to 0, Algorithm 1 line 3. For other warping levels, it is calculated by up-scaling the refined disparity map $\{\widetilde{\boldsymbol{\omega}}\}_{t+1}$ from the coarser level, Algorithm 1 line 5. The pre-estimated disparity map is employed for generating a warped LF, Algorithm 1 line 7. We then calculate the total LF motion tensor $\widehat{\mathbf{J}}_g$ and $\widehat{\mathbf{J}}_G$ from the warped LF (lines 8,9). The refined disparity map $\{\widetilde{\boldsymbol{\omega}}\}_t$ is then calculated by finding the incremental displacement $\{\boldsymbol{\delta\omega}\}_t$ and adding it to $\{\boldsymbol{\omega}\}_t$, Algorithm 1 lines 10-11.

2.3.4 Weighted Median Filtering

Due to occlusion, Eq. 2.5 often fails to capture an accurate disparity value at displacement-edge region. This is due to the fact that occluded pixels may not find their correspondences in all sub-aperture images. Therefore, the constancy assumptions become less reliable and result in the wrong estimation of disparity value at these locations. In order to alleviate this problem, we follow the previous works [81, 82] to apply weighted median filtering to refine the estimated disparity map. Besides helping to improve disparity accuracy, this enhancement step also offers some practical advantages at the architecture level. First, the filter-based operation is well-suited for parallel computing and requires less computation effort than an approach that explicitly models occlusion [30]. Secondly, as an independent processing task, it can also be switched on or off for balancing the output quality and the processing time.

Given a disparity map $\boldsymbol{\omega} : \Omega \longrightarrow \mathbb{R}$ and a non-negative weighting function $w : \Omega \times \mathcal{N} \longrightarrow \mathbb{R}_+, \mathcal{N} \subset \Omega$, the weighted median filter is defined by solving the following equation

$$\widetilde{\boldsymbol{\omega}}_\iota = \arg\min_x \sum_{\tilde{\iota}\in\Omega} w_{\iota,\tilde{\iota}} \left| x - \boldsymbol{\omega}_{\tilde{\iota}} \right| , \tag{2.42}$$

where $\iota, \tilde{\iota} \in \Omega$ denotes the pixel location where the weighted median filtering is applied. The subset $\mathcal{N}$ can span the full domain (i.e., non-local median) or narrow to the neighborhood of ι (i.e., local median) [83]. We employ the later type in this work and define $\mathcal{N}_r(\iota)$ as a square region surrounding ι with the radius of r.

There are many options for the weighting kernel ($w_{\iota,\tilde{\iota}}$). One typical choice is the box kernel which uses constant weights. The box kernel gives us the normal effect of the median filter which is good for removing noise. However, the neighbor is treated in an equal manner which may lead to morphological artifacts and canceling narrow structures [82]. Since we compute the disparity map with respect to the central sub-aperture image, it makes sense to perform post-processing to sharpen the disparity field with respect to this reference view through a guided image kernel as introduced in [84]. For applying the guided median filter, we first construct a binary mask marking occluded areas where there is a high possibility

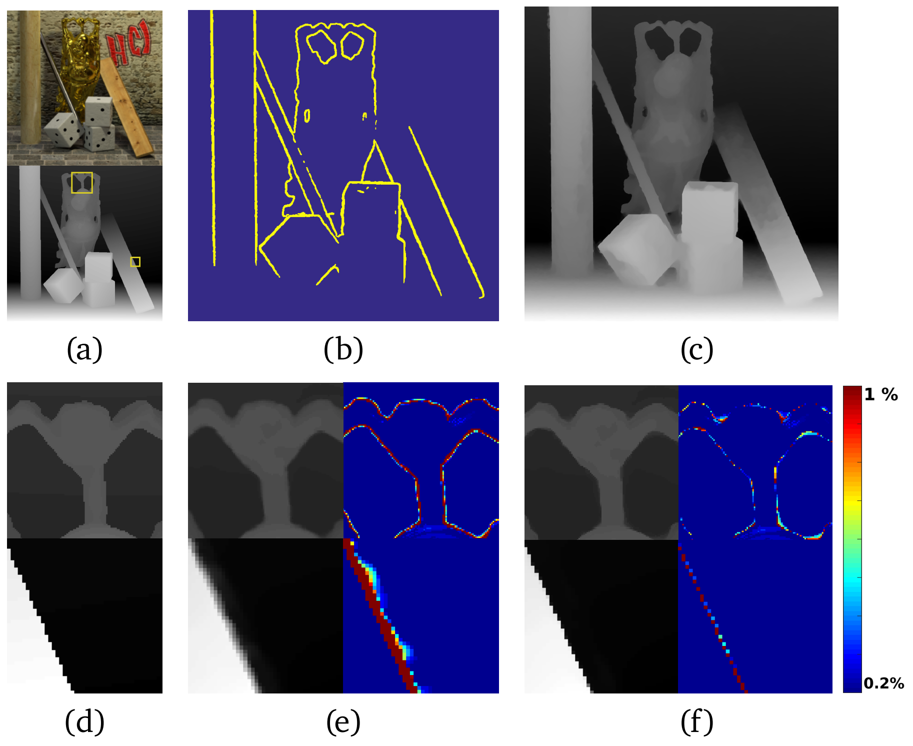


Figure 2.2: Impact of proposed post-processing method on estimated disparity field. The result is computed for *buddha2* scene from Synthetic dataset [14]. (a) *top*: the central sub-aperture image. *bottom*: the ground truth with two interest areas. (b) Estimated occlusion area. (c) Computed disparity map. (d) Zoom in ground truth for two interest areas. (e),(f) estimated disparity and relative error before and after guided filtering respectively.

of erroneous pixels. We then apply the guided median filtering on the disparity map with respect to these occluded areas. We notice that the necessary condition for occlusion is depth discontinuity, and therefore we propose a simple occlusion detection based on the computed displacement field

$$P_{occ} = f(B_r * |\nabla_2 \boldsymbol{\omega}|^2), \tag{2.43}$$

where $f(\cdot)$ is a binary marking function that marks the response above some threshold and B_r* denotes the convolution with a box kernel with size r to expand suspected occlusion areas. An example of marked occluded areas is shown in Figure 2.2(b). The guided median filter with the central sub-aperture image $Ic = L(\mathbf{z}, \boldsymbol{\theta}_0)$ as a guide is applied to the marked area. This allows us to have a sharper and more precise disparity discontinues as could be seen in Figure 2.2(e),(f).

The limitation of guided median filtering is that it only relies on the texture information from the reference SAI. This could introduce ghosting artifact due to the striking change in guided texture which, however, is not reflect the disparity gap. To alleviate this issue, we employ the weighting kernel in [81] in which the weight $w_{\tilde{\iota},\iota}$ presents how likely the two pixels $\tilde{\iota}$ and ι belong to the same depth and is formulated as follow

$$w_{\tilde{\iota},\iota} = \frac{o(\tilde{\iota})}{o(\iota)} e^{\frac{-\|\tilde{\iota}-\iota\|_2^2}{2\sigma_1^2}} e^{\frac{-\|L(\iota,\theta_0)-L(\tilde{\iota},\theta_0)\|_2^2}{2\sigma_2^2}}. \tag{2.44}$$

where $o(\iota)$, defined in Eq. 2.45, is an occlusion weight adopted from [85].

$$o(\mathbf{z}) = e^{-\frac{b(\mathbf{z})^2}{2\sigma_3^2}} e^{-\frac{p(\mathbf{z})^2}{2\sigma_4^2}} \tag{2.45}$$

$b(\mathbf{z})$ and $p(\mathbf{z})$ are occluded boundary function and projection error function respectively. $b(\mathbf{z})$ allows selecting occluding boundary by a one-side divergence.

$$b(\mathbf{z}) = \begin{cases} \mathtt{sum}\{\nabla\boldsymbol{\omega}(\mathbf{z})\}, & \mathtt{sum}\{\nabla\boldsymbol{\omega}(\mathbf{z})\} < 0 \\ 0, & \mathtt{otherwise} \end{cases} \tag{2.46}$$

The projection error function penalizes the accuracy of the constancy assumption with a given disparity $\boldsymbol{\omega}(\mathbf{z})$.

$$p(\mathbf{z}) = \boldsymbol{L}(\mathbf{z}, \boldsymbol{\theta}_0) - \widehat{L}(\mathbf{z}, \boldsymbol{\theta}_i) \tag{2.47}$$

2.3.5 Extension to Multi-channel Light Field

In the previous sections, we presented the application of constancy assumptions and total variation priors in a variational optimization approach to estimate the disparity map. Until this point, the computation has been derived for single-color channel LF. This section discusses an extension to multi-channel LF, i.e., color LF, hyper spectral LF. As compared to single-channel LF, the key difference of multi-channel LF lies in the formulation of the data term which essentially depends on the characteristics of the enclosing channels, which is either correlated or non-correlated. In the case of correlated color channels (i.e., RGB color space), the constancy assumptions of all color channel can be summed up within a robust norm

$$D(\boldsymbol{\omega}) = \Psi\Big(\sum_{\boldsymbol{\theta}_i\in\Pi}\sum_{c\in\{RGB\}} \Big\|\widehat{L}_i^c - \widehat{L}_0^c(\mathbf{z})\Big\|_2^2\Big) + \gamma\Psi\Big(\sum_{\boldsymbol{\theta}_i\in\Pi}\sum_{c\in\{RGB\}} \Big\|\nabla\widehat{L}_i^c - \nabla\widehat{L}_0^c\Big\|_2^2\Big), \tag{2.48}$$

where $\widehat{L_i^c} = \boldsymbol{L}^c(\mathbf{z} + \boldsymbol{\omega}\boldsymbol{\theta}_i, \boldsymbol{\theta}_i)$ denotes the color component c of the warped multi-channel LF. Eventually, this data term leads to the following LF motion tensors

$$\widehat{\mathbf{J}}_g = \sum_{c\in\{R,G,B\}} \widehat{\mathbf{J}}_{g,c} \tag{2.49}$$

$$\widehat{\mathbf{J}}_G = \sum_{c\in\{R,G,B\}} \mathbf{J}_{G,c}, \tag{2.50}$$

with $\mathbf{J}_{g,c}$ and $\mathbf{J}_{G,c}$ are respectively photometric and gradient light field motion tensors of the color channel c. Each type of constancy assumption requires only one robustification function (Ψ) and Eq. 2.40 remains valid.

Regarding non-correlated color channels (i.e., HSV color space), the constancy assumptions of each color channel need to be separately evaluated by robust functions as suggested in [86]

$$D(\boldsymbol{\omega}) = \sum_{c\in\{H,S,V\}} \beta_c \Psi\Big(\sum_{\boldsymbol{\theta}_i\in\Pi} \left\|\widehat{L_i^c} - \widehat{L_0^c}(\mathbf{z})\right\|_2^2\Big) + \gamma_c \sum_{c\in\{H,S,V\}} \Psi\Big(\sum_{\boldsymbol{\theta}_i\in\Pi} \left\|\nabla\widehat{L_i^c} - \nabla\widehat{L_0^c}\right\|_2^2\Big), \tag{2.51}$$

which give use the following outer-inner iterative equation

$$0 = \sum_{c\in\{H,S,V\}} \beta_c \widehat{\Psi}'^{(t,l)}_{g,c} \cdot \left(\widehat{\mathbf{J}}^{(t)}_{g,c,11}\boldsymbol{\delta\omega}^{(t,l+1)} + \widehat{\mathbf{J}}^{(t)}_{g,c,12}\right) + \gamma_c \sum_{c\in\{H,S,V\}} \widehat{\Psi}'^{(t)}_{G,c} \cdot \left(\widehat{\mathbf{J}}^{(t)}_{G,c,11}\boldsymbol{\delta\omega}^{(t,l+1)} + \widehat{\mathbf{J}}^{(t)}_{G,c,12}\right)$$
$$- \alpha\,\mathbf{div}\left(\widehat{\Psi}'^{(t,l)}_s \cdot (\nabla\boldsymbol{\omega}^{(t)} + \nabla\boldsymbol{\delta\omega}^{(t,l+1)})\right),$$

where the derivative $\widehat{\Psi}_*$ and LF motion $\widehat{\mathbf{J}}_*$ tensors are separately computed for each color channels.

2.3.6 Per Perspective Robustification

So far, the robust norm has been applied to a super pixel which is the assemble of SAI pixel from all perspectives. When applying to each perspective pixel separately, we have the following data term

$$D(\boldsymbol{\omega}) = \sum_{\boldsymbol{\theta}_i\in\Pi} \beta_i \Psi\left(\left\|\widehat{L}_i - \widehat{L}_0\right\|_2^2\right) + \sum_{\boldsymbol{\theta}_i\in\Pi} \gamma_i \Psi\left(\left\|\nabla\widehat{L}_i - \nabla\widehat{L}_0\right\|_2^2\right).$$

Here we introduce two sets of parameters β_i and γ_i to weight the contributions of robust norms at each perspective. The main idea is that the constancy assumption may become less reliable for some particular perspectives. For example in plenoptic image, the SAIs at the border normally suffer from distortion and vignetting effects [71]. This data term gives us the following outer-inner iterative equation

$$\begin{aligned} 0 = &\sum_{\theta_i \in \Pi} \beta_i \widehat{\Psi}'^{(t,l)}_{g,i} \cdot \left(\widehat{\mathbf{J}}^{(t)}_{g,i,11} \boldsymbol{\delta\omega}^{(t,l+1)} + \widehat{\mathbf{J}}^{(t)}_{g,i,12}\right) + \gamma_i \sum_{\theta_i \in \Pi} \widehat{\Psi}'^{(t)}_{G,i} \cdot \left(\widehat{\mathbf{J}}^{(t)}_{G,i,11} \boldsymbol{\delta\omega}^{(t,l+1)} + \widehat{\mathbf{J}}^{(t)}_{G,i,12}\right) \\ &- \alpha \mathbf{div} \left(\widehat{\Psi}'^{(t,l)}_{s} \cdot \left(\nabla \boldsymbol{\omega}^{(t)} + \nabla \boldsymbol{\delta\omega}^{(t,l+1)}\right)\right), \end{aligned} \tag{2.52}$$

As compared to Eq. 2.40, it could be seen that the above equation requires much more computational resources. Both the number of computation and the storage size are multiplied by a factor of $s_\rho s_\tau$, where s_ρ and s_τ are the number of perspective in horizontal and vertical directions of LF.

2.4 Numerical Computation

This section discusses the numerical computation to the solution to the problem discussed in the previous sections. Noticed that our derived Euler-Lagrange equations share a similar form to that presented in optic flow literature [74, 75, 77], i.e., except for the number of unknown and the computation of LF motion tensor which are formulated for 4DLF disparity estimation problem. Therefore, the numerical computation, presented in this section, inherited largely from these works, especially from the lecture notes[87].

2.4.1 Discreatization Strategy and Notation

The proposed variational model consists of two dimensional functions defined on the spatial coordinate Ω, i.e., $f(\mathbf{z}) : \Omega \rightarrow \mathbb{R}, \quad \mathbf{z} = [x \quad y]^\intercal \in \Omega$. Although treated as continuous functions, the numerical computation needs to be carried out on their discretization basis. For discretization, we define a two-dimensional regular Cartesian grid of the size $H \times W$, where H is the height and W is the width of the sub-aperture image

$$\{(i\Delta_x, j\Delta_y) | 1 \leq i \leq W, 1 \leq j \leq H\},$$

with the indices of the discrete coordinate given by (i, j) and the step sizes in x and y direction are Δ_x and Δ_y, respectively. Discretized functions, therefore, belong to a finite dimensional vector space $\overline{\Omega} = \mathbb{R}^{H\times W}$ with the scalar product defined as

$$< U, V >= \sum_{i,j} V_{i,j} U_{i,j}, \quad U, V \in \overline{\Omega}.$$

For the computation of the first-order derivative, we apply the central difference

$$\left[\frac{\partial}{\partial x} X\right]_{i,j} = \frac{[X]_{i+1,j} - [X]_{i-1,j}}{2\Delta_x},$$

and the second order derivative

$$\left[\frac{\partial}{\partial^2 x} X\right]_{i,j} = \frac{[\frac{\partial}{\partial x}X]_{i+\frac{1}{2},j} - [\frac{\partial}{\partial x}X]_{i-\frac{1}{2},j}}{\Delta_x} = \frac{\frac{[X]_{i+1,j}-[X]_{i,j}]}{\Delta_x} - \frac{[X]_{i,j}-[X]_{i-1,j}}{\Delta_x}}{\Delta_x} = \frac{[X]_{i+1,j}+[X]_{i-1,j}-2[X]_{i,j}}{\Delta_x^2}.$$

The derivative across the boundary is set to 0, i.e., vanished as follow reflecting Neumann boundary condition.

As forming a system of linear equations would require the column-vector representation of discretized functions, let us define a linear mapping function Φ

$$\begin{aligned} \Phi : \quad & \mathbb{R}^{W\times H} \longrightarrow \mathbb{R}^{WH} \\ & X \longrightarrow \mathbf{x}, [\mathbf{x}]_k = [X]_{i,j}, k = iH + j, \end{aligned} \tag{2.53}$$

which performs a vectorization on a 2D array input. Since this mapping function is an isomorphism, there exists an inverse mapping Φ^{-1} that reverts the column vector into its 2D form. Besides, the properties of a linear mapping function, Φ has also the following property applied to Hadamard operators

$$\begin{aligned} \Phi(A \odot B) = \quad & \Phi(A) \odot \Phi(B), \\ \Phi(A \oslash B) = \quad & \Phi(A) \oslash \Phi(B), \end{aligned} \tag{2.54}$$

where $\odot$ and $\oslash$ denote the Hadamard product and Hadamard division respectively. Let us define a shifting map $\Xi_{i,j}$ which respectively shifts the image i pixel and j pixel horizontally and vertically.

$$\begin{aligned} \Xi_{\delta_i,\delta_j} : \quad & \mathbb{R}^{W\times H} \longrightarrow \mathbb{R}^{W\times H} \\ & X \longrightarrow Y, \quad [Y]_{i,j} = [X]_{i+\delta_i, j+\delta_j}, \end{aligned} \tag{2.55}$$

which is equal to applying a matrix transformation on a vectorization

$$\Phi\left(\Xi_{\delta_i,\delta_j}(X)\right) = S_{\delta_i,\delta_j}\Phi(X).$$

2.4.2 Generalized Equation

As will be shown in the next section, the Euler-Lagrange equations derived in the previous sections share the following form

$$\begin{aligned} 0 = \; & [M_1]_{i,j}[X]_{i,j} + [M_2]_{i,j} \\ & + [P_1]_{i,j}\left([X]_{i+1,j} - [X]_{i,j}\right) + [P_2]_{i,j}\left([X]_{i-1,j} - [X]_{i,j}\right) \\ & + [Q_1]_{i,j}\left([X]_{i,j+1} - [X]_{i,j}\right) + [Q_2]_{i,j}\left([X]_{i,j-1} - [X]_{i,j}\right), \end{aligned} \tag{2.56}$$

where matrices $M_*, Q_*, P_*, X \in \mathbb{R}^{m\times n}$ and $[\cdot]_{i,j}$ denotes the element at i^{th} row and j^{th} col. While M_*, Q_*, P_* are constant, the matrix X is an unknown that we are going to estimate. Assembling Eq. 2.56 of all possible indices (i, j), we obtain a system of linear equations taking the form

$$A\mathbf{x} = \mathbf{b},$$

where $A \in \mathbb{R}^{mn\times mn}$, and $\mathbf{x}, \mathbf{b} \in \mathbb{R}^{mn}$. In order to derive the structure of the sparse matrix A and column vector $\mathbf{b}$, let us rewrite Eq. 2.56 in vector form

$$\begin{aligned} 0 = \; & \Phi(M_1) \odot \mathbf{x} + \Phi(M_2) \\ & + \Phi(P_1) \odot \left(S_{1,0}\mathbf{x} - \mathbf{x}\right) + \Phi(P_2) \odot \left(S_{-1,0}\mathbf{x} - \mathbf{x}\right) \\ & + \Phi(Q_1) \odot \left(S_{0,1}\mathbf{x} - \mathbf{x}\right) + \Phi(Q_2) \odot \left(S_{0,-1}\mathbf{x} - \mathbf{x}\right), \end{aligned} \tag{2.57}$$

where $\mathbf{x} = \Phi(X)$ and vectorization function $\Phi : \mathbb{R}^{m\times n} \longrightarrow \mathbb{R}^{mn}$ simply forms a column vector $\mathbf{x}$ by stacking all columns of matrix X. The transformation matrix $S_{\delta_i,\delta_j} \in \mathbb{R}^{mn\times mn}$ shifts the matrix X by δ_i and δ_j pixels in x and y direction, respectively. From Eq. 2.57, it is obvious that

$$\begin{aligned} \mathbf{b} = \; & -\Phi(M_2) \\ A = \; & \operatorname{diag}\big(\Phi(M_1) - \Phi(P_1) - \Phi(P_2) - \Phi(Q_1) - \Phi(Q_2)\big) + \operatorname{diag}\big(\Phi(P_1)\big)S_{1,0} \\ & + \operatorname{diag}\big(\Phi(Q_1)\big)S_{0,1} + \operatorname{diag}\big(\Phi(P_2)\big)S_{-1,0} + \operatorname{diag}\big(\Phi(Q_2)\big)S_{0,-1}, \end{aligned} \tag{2.58}$$

where $\operatorname{diag}(\mathbf{y})$ denotes the diagonal matrix constructed from the column vector $\mathbf{y}$, i.e., $[\operatorname{diag}(\mathbf{y})]_{i,i} = [\mathbf{y}]_i$. To employ iterative solvers such as Jacobi and Gauss-Seidel for solving the sparse system of linear equations, we first need to decompose the sparse matrix A into three parts

$$A = D + U + L,$$

where D, U, and L denote the diagonal component, upper triangular part, and lower triangular part. From the construction of A in Eq. 2.58, we have

$$\begin{aligned} D =& \ \operatorname{diag}\big(\Phi(M_1) - \Phi(P_1) - \Phi(P_2) - \Phi(Q_1) - \Phi(Q_2)\big) \\ U =& \ \operatorname{diag}\big(\Phi(P_1)\big)S_{1,0} + \operatorname{diag}\big(\Phi(Q_1)\big)S_{0,1} \\ L =& \ \operatorname{diag}\big(\Phi(P_2)\big)S_{-1,0} + \operatorname{diag}\big(\Phi(Q_2)\big)S_{0,-1}. \end{aligned} \tag{2.59}$$

From this point, the Jacobi iterative method can be applied to solve the sparse system of linear equations. The iterative step for finding the solution $\mathbf{x}$ is as follows

$$\mathbf{x}^{(k+1)} = D^{-1}\left(\mathbf{b} - (L+U)\mathbf{x}^{(k)}\right).$$

To avoid the sparse matrix implementation, we are going to derive element-wise numerical computation. Let us first revert the above equation back to its matrix form

$$\Phi^{-1}(\mathbf{x}^{(k+1)}) = \Phi^{-1}(D^{-1}) \odot \left(\Phi^{-1}(\mathbf{b}) - \Phi^{-1}\big(L\mathbf{x}^{(k)}\big) - \Phi^{-1}\big(U\mathbf{x}^{(k)}\big)\right), \tag{2.60}$$

where $\Phi^{-1}\left(\mathbf{x}^{(k+1)}\right) = X^{(k+1)}$, $\Phi^{-1}\left(D^{-1}\right) = 1 \oslash (M_1 - P_1 - P_2 - Q_1 - Q_2)$, $\Phi^{-1}(\mathbf{b}) = -M_2$, and

$$\begin{aligned} \Phi^{(-1)}\big(U\mathbf{x}\big) =& P_1 \odot \Phi^{-1}(S_{1,0}\mathbf{x}) + Q_1 \odot \Phi^{-1}(S_{0,1}\mathbf{x}), \\ \Phi^{(-1)}\big(L\mathbf{x}\big) =& P_2 \odot \Phi^{-1}(S_{-1,0}\mathbf{x}) + Q_2 \odot \Phi^{-1}(S_{0,-1}\mathbf{x}), \end{aligned}$$

with $\Phi^{-1}(S_{\delta_i,\delta_j}\mathbf{x}) = \Xi_{\delta_i,\delta_j}(X)$. We rewrite Eq. 2.60 as

$$\begin{aligned} X^{(k+1)} =& \text{-}\Big(M_2 + P_1 \odot \Xi_{1,0}(X^{(k)}) + P_2 \odot \Xi_{-1,0}(^{(k)}X) + Q_1 \odot \Xi_{0,1}(X^{(k)}) + Q_2 \odot \Xi_{0,-1}(X^{(k)})\Big) \\ & \oslash \big(M_1 - P_1 - P_2 - Q_1 - Q_2\big), \end{aligned} \tag{2.61}$$

which allows us to form this element-wise computation

$$[X]_{i,j}^{(k+1)} = \frac{-[M_2]_{i,j} - [P_1]_{i,j}[X]_{i+1,j}^{(k)} - [P_2]_{i,j}[X]_{i-1,j}^{(k)} - [Q_1]_{i,j}[X]_{i,j+1}^{(k)} - [Q_2]_{i,j}[X]_{i,j-1}^{(k)}}{[M_1]_{i,j} - \big([P_1]_{i,j} + [P_2]_{i,j} + [Q_1]_{i,j} + [Q_2]_{i,j}\big)} \tag{2.62}$$

For Gauss-Seidel method, the iterative step for finding the solution $\mathbf{x}$ is

$$\mathbf{x}^{(k+1)} = (L+D)^{-1}\left(\mathbf{b} - U\mathbf{x}^{(k)}\right).$$

Taking advantage of the triangular from of $(L+D)$, the forward substitution technique can be used to sequentially compute $(L+D)^{-1}$ leading to the iterative computation of $\mathbf{x}$

$$\mathbf{x}^{(k+1)} = D^{-1}(b - U\mathbf{x}^{(k)} - L\mathbf{x}^{(k+1)}) \tag{2.63}$$

In Eq. 2.63, it is important that the elements of $\mathbf{x}$ are computed with a strict order, i.e., starting from the first element and one after another. Following the similar transformation as of Jacobi method, an element-wise computation of X is as follows

$$[X]_{i,j}^{(k+1)} = \frac{-[M_2]_{i,j} - [P_1]_{i,j}[X]_{i+1,j}^{(k)} - [P_2]_{i,j}[X]_{i-1,j}^{(k+1)} - [Q_1]_{i,j}[X]_{i,j+1}^{(k)} - [Q_2]_{i,j}[X]_{i,j-1}^{(k+1)}}{[M_1]_{i,j} - \left([P_1]_{i,j} + [P_2]_{i,j} + [Q_1]_{i,j} + [Q_2]_{i,j}\right)}$$

2.4.3 Solution to Quadratic Data Term and Smoothness Term

This section discusses the application of the proposed numerical computation to the variational problem discussed in Sec. 2.3.1. Let us rewrite the Euler-Lagrange equation, Eq. 2.22,

$$0 = [\mathbf{J}_{11}]_{i,j}[\boldsymbol{\omega}]_{i,j} + [\mathbf{J}_{12}]_{i,j} - \alpha\left([\boldsymbol{\omega}_{xx}]_{i,j} + [\boldsymbol{\omega}_{yy}]_{i,j}\right), \tag{2.64}$$

in which the LF tensor components ($\mathbf{J}_*$), disparity map ($\boldsymbol{\omega}$) and derivative terms ($\boldsymbol{\omega}_*$) are treated as 2D array matrices in $\mathbb{R}^{H\times W}$. The divergent term $\mathbf{div}\,\nabla\boldsymbol{\omega} = \boldsymbol{\omega}_{xx} + \boldsymbol{\omega}_{yy}$, with $\boldsymbol{\omega}_{xx} = \frac{\partial}{\partial^2 x}\boldsymbol{\omega}(x,y)$ and $\boldsymbol{\omega}_{yy} = \frac{\partial}{\partial^2 y}\boldsymbol{\omega}(x,y)$ are computed as follow

$$\begin{aligned}
[\boldsymbol{\omega}_{xx}]_{i,j} &= \frac{[\boldsymbol{\omega}_x]_{i+\frac{1}{2},j} \text{-} [\boldsymbol{\omega}_x]_{i-\frac{1}{2},j}}{\Delta_x} = \frac{\frac{[\boldsymbol{\omega}]_{i+1,j}\text{-}[\boldsymbol{\omega}]_{i,j}}{\Delta_x} \text{-} \frac{[\boldsymbol{\omega}]_{i,j}\text{-}[\boldsymbol{\omega}]_{i-1,j}}{\Delta_x}}{\Delta_x} = \frac{[\boldsymbol{\omega}]_{i+1,j} + [\boldsymbol{\omega}]_{i-1,j}\text{-}2[\boldsymbol{\omega}]_{i,j}}{\Delta_x^2} \\
[\boldsymbol{\omega}_{yy}]_{i,j} &= \frac{[\boldsymbol{\omega}_y]_{i,j+\frac{1}{2}} \text{-} [\boldsymbol{\omega}_y]_{i,j-\frac{1}{2}}}{\Delta_y} = \frac{\frac{[\boldsymbol{\omega}]_{i,j+1}\text{-}[\boldsymbol{\omega}]_{i,j}}{\Delta_y} \text{-} \frac{[\boldsymbol{\omega}]_{i,j}\text{-}[\boldsymbol{\omega}]_{i,j-1}}{\Delta_y}}{\Delta_y} = \frac{[\boldsymbol{\omega}]_{i+1,j} + [\boldsymbol{\omega}]_{i-1,j}\text{-}2[\boldsymbol{\omega}]_{i,j}}{\Delta_y^2}
\end{aligned} \tag{2.65}$$

Here a half-step central differential for computing the second order derivative of $\boldsymbol{\omega}$ is applied. This allows us to form the following

$$\begin{aligned}0 =&[\mathbf{J}_{11}]_{i,j}[\boldsymbol{\omega}]_{i,j} + [\mathbf{J}_{12}]_{i,j} - \frac{\alpha}{\Delta_x^2}\left([\boldsymbol{\omega}]_{i+1,j} - [\boldsymbol{\omega}]_{i,j}\right) - \frac{\alpha}{\Delta_x^2}\left([\boldsymbol{\omega}]_{i-1,j} - [\boldsymbol{\omega}]_{i,j}\right)\\ &- \frac{\alpha}{\Delta_y^2}\left([\boldsymbol{\omega}]_{i,j+1} - [\boldsymbol{\omega}]_{i,j}\right) - \frac{\alpha}{\Delta_y^2}\left([\boldsymbol{\omega}]_{i,j-1} - [\boldsymbol{\omega}]_{i,j}\right)\end{aligned} \tag{2.66}$$

Mapping it to Eq. 2.56 gives us the matrices X, M_*,P_*, and Q_*

$$\begin{aligned}X =&[\boldsymbol{\omega}], \quad M_1 = [\mathbf{J}_{11}], \quad M_2 = [\mathbf{J}_{12}],\\ P_1 =& - \frac{\alpha}{\Delta_x^2}[I], \quad P_2 = -\frac{\alpha}{\Delta_x^2}[I], \quad Q_1 = -\frac{\alpha}{\Delta_y^2}[I], \quad Q_2 = -\frac{\alpha}{\Delta_y^2}[I],\end{aligned} \tag{2.67}$$

where $[I] \in \mathbb{R}^{H\times W}$ denotes a matrix filling with one, i.e., $[I]_{i,j} = 1,\ i \in [1, W], j \in [1, H]$.

2.4.4 Solution to Sub-quadratic Data Term and Smoothness term

Let us rewrite Eq. 2.40 considering the inter loop at the iteration l+1th

$$\begin{aligned}0 =\ & \widehat{\Psi}_g'^{(l)} \cdot \left(\widehat{\mathbf{J}}_{g,11}\boldsymbol{\delta\omega}^{(l+1)} + \widehat{\mathbf{J}}_{g,12}\right) + \gamma\widehat{\Psi}_G'^{(l)} \cdot \left(\widehat{\mathbf{J}}_{G,11}\boldsymbol{\delta\omega}^{(l+1)} + \widehat{\mathbf{J}}_{G,12}\right)\\ &- \alpha\mathbf{div}\left(\widehat{\Psi}_s'^{(l)} \cdot \left(\nabla\boldsymbol{\omega} + \nabla\boldsymbol{\delta\omega}^{(l+1)}\right)\right),\end{aligned} \tag{2.68}$$

which has the following discretized form

$$\begin{aligned}0 =& \left([\widehat{\Psi}_g']_{i,j}^{(l)}[\widehat{\mathbf{J}}_{g,11}]_{i,j} + \gamma[\widehat{\Psi}_G']_{i,j}^{(l)}[\widehat{\mathbf{J}}_{G,11}]_{i,j}\right)[\boldsymbol{\delta\omega}]_{i,j}^{(l+1)}\\ &+ \left([\widehat{\Psi}_g']_{i,j}^{(l)}[\widehat{\mathbf{J}}_{g,12}]_{i,j} + \gamma[\widehat{\Psi}_G']_{i,j}^{(l)}[\widehat{\mathbf{J}}_{G,12}]_{i,j}\right)\\ &- \alpha[\mathbf{div}\left(\widehat{\Psi}_s'^{(l)}\nabla\boldsymbol{\omega}\right)]_{i,j} - \alpha[\mathbf{div}\left(\widehat{\Psi}_s'^{(l)}\nabla\boldsymbol{\delta\omega}^{(l+1)}\right)]_{i,j},\end{aligned} \tag{2.69}$$

It follows that

$$\begin{aligned}X =\ & [\boldsymbol{\delta\omega}]^{(l+1)}\\ M_1 =\ & [\widehat{\Psi}_g']^{(l)} \odot [\widehat{\mathbf{J}}_{g,11}] + \gamma[\widehat{\Psi}_G']^{(l)} \odot [\widehat{\mathbf{J}}_{G,11}]\\ M_2 =\ & [\widehat{\Psi}_g']^{(l)} \odot [\widehat{\mathbf{J}}_{g,12}] + \gamma[\widehat{\Psi}_G']^{(l)} \odot [\widehat{\mathbf{J}}_{G,12}] + \alpha[\mathbf{div}\left(\widehat{\Psi}_s'^{(l)}\nabla\boldsymbol{\omega}\right)].\end{aligned} \tag{2.70}$$

Applying a similar strategy as in previous section, the divergent term is computed as follows

$$[\mathbf{div}\,(\widehat{\Psi}_s'^{(l)}\nabla\boldsymbol{\omega})]_{i,j} = \sum_{\eta\in\{x,y\}}\sum_{(u,v)\in\mathcal{N}_\eta}\frac{[\widehat{\Psi}_s']^{(l)}_{i+u,j+v}+[\widehat{\Psi}_s']^{(l)}_{i,j}}{2}\frac{[\boldsymbol{\omega}]_{i+u,j+v}-[\boldsymbol{\omega}]_{i,j}}{\Delta_\eta^2}, \tag{2.71}$$

where $\mathcal{N}_\eta, \eta \in \{x,y\}$ signifies the direct neighbor set, $\mathcal{N}_x = \{(1,0),(-1,0)\}$ and $\mathcal{N}_y = \{(0,1),(0,-1)\}$ follows the notation in [77]. In the same manner, the divergent term of incremental amount $\boldsymbol{\delta\omega}$ is computed as

$$[\mathbf{div}\,(\hat{\Psi}_s'^{(l)}\nabla\boldsymbol{\delta\omega}^{(l+1)})]_{i,j} = \sum_{\eta\in\{x,y\}}\sum_{(u,v)\in N_\eta}\frac{[\hat{\Psi}_s']^{(l)}_{i+u,j+v}+[\hat{\Psi}_s']^{(l)}_{i,j}}{2}\frac{[\boldsymbol{\delta\omega}]^{(l+1)}_{i+u,j+v}-[\boldsymbol{\delta\omega}]^{(l+1)}_{i,j}}{\Delta_\eta^2}. \tag{2.72}$$

The matrices P_* and Q_* can be derived as

$$\begin{aligned} P_1 &= \frac{[\hat{\Psi}_s']^{(l)}_{i+1,j}+[\hat{\Psi}_s']^{(l)}_{i,j}}{2\Delta_x^2}, \quad P_2 = \frac{[\hat{\Psi}_s']^{(l)}_{i-1,j}+[\hat{\Psi}_s']^{(l)}_{i,j}}{2\Delta_x^2} \\ Q_1 &= \frac{[\hat{\Psi}_s']^{(l)}_{i,j+1}+[\hat{\Psi}_s']^{(l)}_{i,j}}{2\Delta_y^2}, \quad Q_2 = \frac{[\hat{\Psi}_s']^{(l)}_{i,j-1}+[\hat{\Psi}_s']^{(l)}_{i,j}}{2\Delta_y^2} \end{aligned} \tag{2.73}$$

2.5 Experimental Results

The performance of the proposed variational framework was evaluated using both synthetic and real-world datasets. The synthetic 4D Light field dataset [14] was used for quantitative comparisons with related work. All the real-world dataset was captured using lenslet-based light field camera Lytro. We used both light field data provided by EPFL [18] and data captured by our Lytro Ilium camera.

We implemented robustification model,i.e., Eq. 2.23, in MATLAB. The computation ran on Intel i5 2.4Ghz CPU with 8GB Ram and required 4 minutes for *Lytro* dataset and 9 minutes for the Synthetic dataset. The parameters $\alpha, \beta_*, \gamma_*$ are adjusted for each light field data. The other parameters were selected as $l = 11$, $\eta = 0.8$, $\sigma = 0.5$, $\epsilon = 0.001^2$. We notice that the most time consuming tasks are warping ($\widehat{\boldsymbol{L}}_i = \boldsymbol{L}_i(\mathbf{z}+\boldsymbol{\theta}_i\boldsymbol{\omega})$) and computing light field motion tensor $(\widehat{\mathbf{J}}_g, \widehat{\mathbf{J}}_g)$. These tasks could be dramatically speeded up on high-parallelism platforms such as GPU or FPGA.

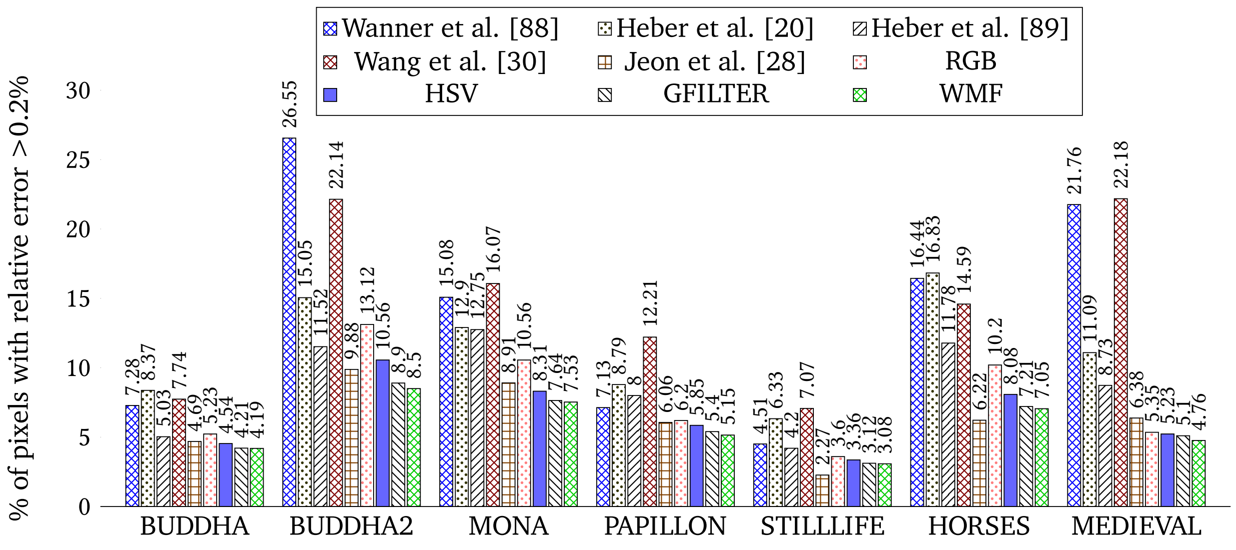


Figure 2.3: Relative error comparison on synthetic dataset [14]

The relative error is adopted as a common metric for comparison. For each synthetic light field data, we computed the percentage of pixels with a relative depth error of more than 0.2%. As mentioned by Heber [20], this is the smallest meaningful accuracy level since the depth discretization of the provided ground truth is too low. We reported the result for both RGB and HSV color space and the post-processing result of the best solution.

Figure 2.3 shows the relative error results of the previous works and ours. It could be seen from the bar chart that our proposed framework outperforms previous continuous modelling approaches [20,88,89] in term of accuracy. Compared with the RGB color space, the computation on the HSV color space provides more precision due to the separate robustification of the data term. The guided median filtering (GFILTER) and weighted median filtering (WMF) further improve the accuracy by sharpening the disparity discontinues. Compared to GFILTER which employs only texture information from the reference SAI to sharpen the disparity map, WMF is more robust due to the incorporation of occlusion weights leading to a slightly improvement in the accuracy. We notice that the approach of Wang et al. [30] is very sensitive to estimated occlusion results, and tends to have a wrong disparity estimation at strong texture areas. Their approach therefore is less accurate when compared to the others.

Compared with the work of Jeon et al. [28], our approach provides better results for the five out of seven synthetic light field data, while scores less for the two challenging scenes ’STILLLIFE’ and ’HORSES’. We noticed that these scenes contains a large amount of depth discontinues in which our constancy assumptions does not hold true. This may suggest

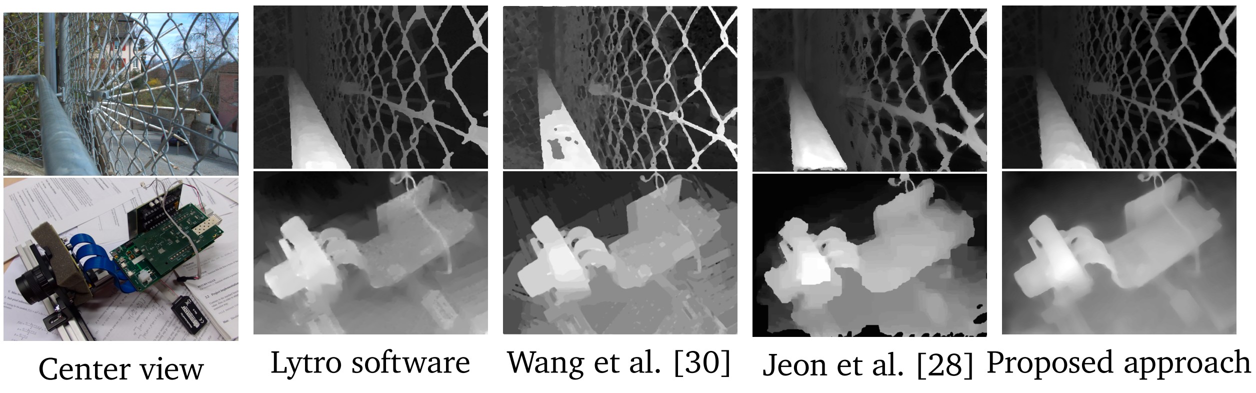


Figure 2.4: Comparison on real-world dataset

a future work on data terms which explicitly model occlusion for 4DLF image. However, there are some advantages of our approach that worth mentioning. First, our approach does not require previous knowledge of disparity range as well as spatial disparity unit for each disparity label due to the continuous formation. Without this knowledge, discrete label-based approaches [30],[28] need to increase the number of labels and reduce the spatial unit in order to sufficiently cover the disparity range. It consequently increases the computation time. In addition, a number of well-turned parameters for each data and each processing steps are critical in their approach [28] in order to guarantee a smooth disparity field without outliers. Our approach, in contrary, relies mainly on weighting of data and smoothness term.

As qualitative comparison, the results on real-world dataset obtained by using *Lytro* camera is presented. The images are captured using Ilium version which provides a spatial resolution of around 434×625 and a directional resolution of about 15×15. Due to the vignetting impact of microlens, the effective directional resolution is limited to only 193 views (85.7%). Figure 2.5 presents the results of our proposed approach for different scenes. Along with the central sub-aperture image, there is also a rendered 3D view as well as a color-coded disparity map. Figure 2.4 shows the disparity map for two different scenes computed by the Lytro software, the methods of Wang et al.[30], Jeon et al. [28] and ours. The results for [30],[28] are computed using their provided codes. The figure shows that our proposed approach provides solutions with more details and fewer artifacts.

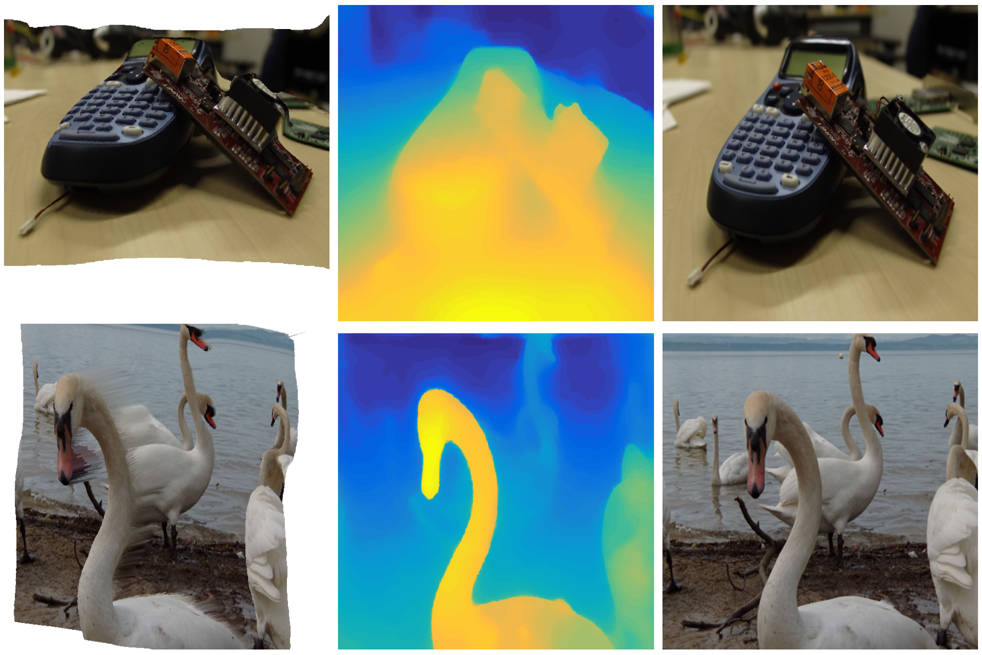


Figure 2.5: Real-world result. *top* scene from our laboratory captured light field. *bottom* scene from EPFL dataset [18]

2.6 Summary

In this chapter, we presented a variational computation approach for the 4DLF disparity estimation problem. Coarse-to-fine warping scheme was employed to alleviate the invalidity of linearization at large displacement area. Weighted median filtering was employed as a post-processing technique to further enhance the accuracy at occluded area. The numerical computation derived for iterative solving of the disparity maps provides a detailed information for further development of the acceleration approach. The experimental results demonstrate our capability of computing accurate disparity solution on both challenging synthetic and real-world light field dataset.

Chapter 3

GPU-Accelerated 4D Light Field Disparity Estimation

As discussed in the previous chapter, solving the minimization problem to estimate the disparity maps of 4DLF images involves a nested iteration over multi-scale 4D data, which indeed demands a high computational effort. To alleviate the high computational requirement, this chapter presents a GPU-accelerated approach in which the time-consuming computing tasks are offloaded to GPU. This results in a significant speed-up of disparity estimation. The chapter starts with an overview to the OpenCL computing framework and its application to 4DLF disparity estimation. The architecture of the proposed accelerator with regard to the numerical computation presented in Chapter 2 is discussed in the next section. In Sec. 3.3.1, an extensive evaluation and detailed analysis of the GPU realization are reported. Parts of the results of this chapter have been published in [21].

3.1 OpenCL-based 4DLF Disparity Estimation

The acceleration is achieved by the mean of parallel computation using GPUs and the OpenCL framework. For a better understanding of our proposed architecture choices, terminologies and basic properties of the OpenCL framework are first overviewed.

In OpenCL, a computing system is considered a collection of many *compute devices* such as CPUs, GPUs, DSPs, FPGAs, and other accelerators. Each compute device typically consists of a number of *compute units*, which in turn comprises multiple *compute elements*. A programming workload in OpenCL is divided into *work-groups* and *work-items* which are executed on compute units and compute elements respectively. The maximum number of work items constituting a workgroup is device-dependent. Each work-group is executed independently with respect to the other work-groups, while work-items within a work-group are more or less dependent due to the fact that they are executed in a thread-based manner and on the same compute unit. Each work-item executes a set of C-like instructions organized as a function, also known as a *kernel*.

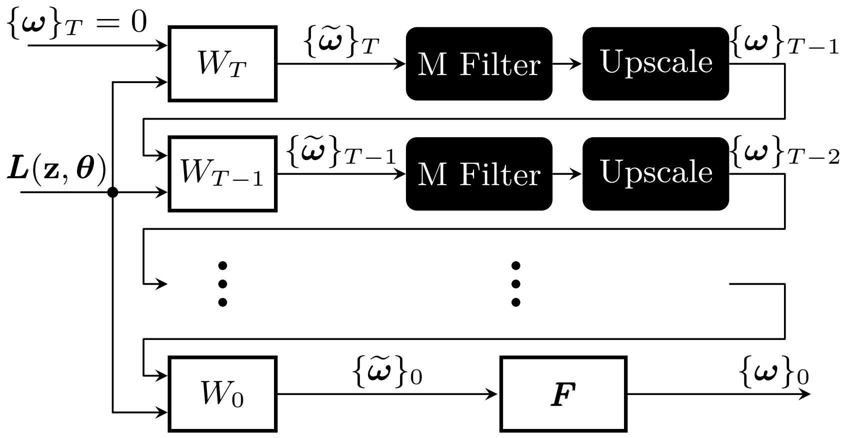
OpenCL bases on the Host-Device model in which the computing program is loaded and started on a host and then some or most of the programming workload might be off-loaded to devices. On each device, the memory model distinguishes different memory regions: global, local, constant, and private memory. Global memory is read-write accessible by all work-items across work-groups. Local memory is specified to a work-group and is only accessible to work-items located in the same work-group. Constant memory is read-only and has the same access-range as global memory. Private memory is specified to a work-item and therefore not visible to other work-items. Global memory and local memory are generally mapped to off-chip memory on compute device and on-chip memory on compute unit respectively. Therefore, global memory is generally more plentiful and requires longer access latency compared to local memory. All constant are first stored on global memory and then cached in constant cache. It would cost the same latency as accessing global memory for the first time. Variables defined in private memory will be mapped to registers located at compute units. If there are not enough registers, it will be mapped to global memory.

In general, there are three main problems which need to be addressed to increase the overall performance: optimizing the number of memory transfers, reducing overall memory access time, and parallelizing accelerated algorithm. Memory transfer is one of the main limitations of the Host-Device model in which a large amount of data is transferred between the host's main memory and devices' global memory. This operation is typically carried out over the PCI Express bus which is with high latency and is also the bottle-neck of the whole program workload. In the proposed accelerator, the number of memory transfers is minimized to two which occur at the beginning and the end of the computation. The first transfer is for sending 4D LF data and algorithm's parameters to GPUs' global memory and the last transfer is for copying back the estimated disparity map from GPUs' global memory to the host's memory. Important factors that affect the overall memory access time are the type of memory, memory access pattern, and data caching. Different memory types have different access latencies. Global memory requires the longest access time, then comes cached constant memory, local memory, and private memory (registers). A good practice applied in this work is maximizing the usage of memory in a higher hierarchy.

Memory access patterns and data caching are also considered and will be discussed further in Section 3.2.2. In the following sections, the parallelization of the disparity estimation algorithm will be discussed in detail. The algorithm is first decomposed into a set of processing functions. These functions are then analyzed to identify their capability of parallel computation. Based on this, the global work size of each GPU kernel is then decided. In the end, all the functions are transformed into OpenCL kernel code and executed for GPU devices.

3.2 GPU-Accelerated Architecture

Figure 3.1 presents the overall computation flow of the proposed approach. As discussed in the previous chapter, warping and coarse-to-fine strategy are employed to handle large disparities in our algorithm. Instead of directly estimating a disparity map $\boldsymbol{\omega}$ of a 4D light field $\boldsymbol{L}(\mathbf{z}, \boldsymbol{\theta})$, the input light field is spatially down-sampled to get a coarser 4D light field $\{\boldsymbol{L}(\mathbf{z}, \boldsymbol{\omega})\}_t$ for which an associated disparity map $\{\boldsymbol{\omega}\}_t$ is then estimated. At each level t ($t = 0, 1, .., T$), the spatial resolution of $\{\boldsymbol{L}(\mathbf{z}, \boldsymbol{\omega})\}_t$ is ζ^t times smaller than of the input light field. For example, if n_y, n_x are the height and the width of one sub-aperture image in the input 4D light field, the height and the width of downsampled light field at level t will be $\lfloor n_y \zeta^t \rfloor$ and $\lfloor n_x \zeta^t \rfloor$ respectively. Initially, the disparity map $\{\boldsymbol{\omega}\}_T$ is set to zero and fed to *coarse-to-fine computation* block (W_t). At the end of each coarse-to-fine level t, $t \neq 0$, we apply median filter (*'M filter'*) on the estimated disparity map $\{\widetilde{\boldsymbol{\omega}}_t\}_t$ to remove noise and upscale it by a factor of ζ^{-1}. This upscaled disparity map is then inputted to the next warping level (W_{t-1}) as demonstrated in Figure 3.1. The weighted median filter



$\{\boldsymbol{\omega}\}_T = 0$
W_T
$\{\widetilde{\boldsymbol{\omega}}\}_T$
M Filter
Upscale
$\{\boldsymbol{\omega}\}_{T-1}$
$\boldsymbol{L}(\mathbf{z}, \boldsymbol{\theta})$
W_{T-1}
$\{\widetilde{\boldsymbol{\omega}}\}_{T-1}$
M Filter
Upscale
$\{\boldsymbol{\omega}\}_{T-2}$
W_0
$\{\widetilde{\boldsymbol{\omega}}\}_0$
$\boldsymbol{F}$
$\{\boldsymbol{\omega}\}_0$

Figure 3.1: Warping architecture

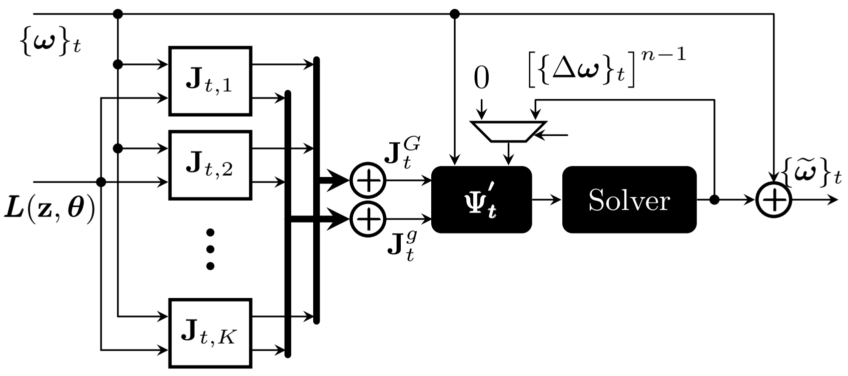
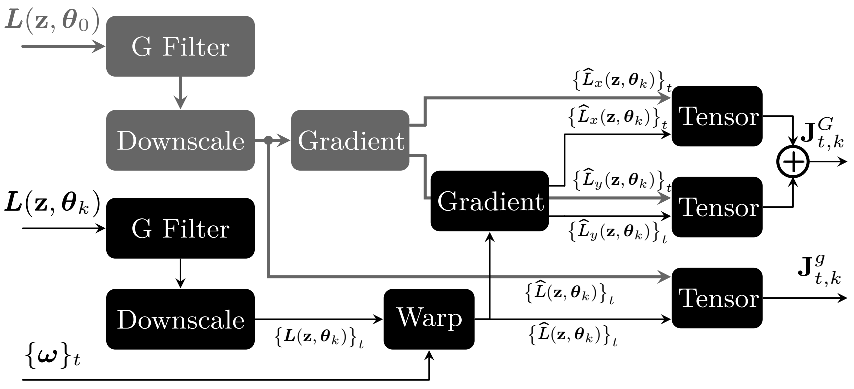


Figure 3.2: Block diagram representation of a single warping level $\boldsymbol{W}_l$



$L(\mathbf{z},\theta_0)$
G Filter
Downscale
Gradient
$L(\mathbf{z},\theta_k)$
G Filter
Downscale
Warp
Gradient
$\{\omega\}_t$
$\{L(\mathbf{z},\theta_k)\}_t$
$\{\widehat{L}_x(\mathbf{z},\theta_k)\}_t$
$\{\widehat{L}_x(\mathbf{z},\theta_k)\}_t$
$\{\widehat{L}_y(\mathbf{z},\theta_k)\}_t$
$\{\widehat{L}_y(\mathbf{z},\theta_k)\}_t$
$\{\widehat{L}(\mathbf{z},\theta_k)\}_t$
$\{\widehat{L}(\mathbf{z},\theta_k)\}_t$
Tensor
Tensor
Tensor
$\mathbf{J}_{t,k}^G$
$\mathbf{J}_{t,k}^g$

Figure 3.3: Block diagram representation of a tensor calucation module $\mathbf{J}_{l,k}$

($\mathfrak{F}$) is applied to the output of the last level, the finest level, to improve the quality of the estimated disparity map.

Figure 3.2 is the block diagram representation of a single coarse-to-fine level (W_t) which includes the motion tensor calculation unit $\mathbf{J}_{t,k}$ ($k = 1, .., K$), robust-weight updating unit (Ψ_t'), and a Jacobian-based iterative solver. The number K denotes the total number of sub-aperture images in the input light field used for calculating the disparity map. Let n_ρ and n_τ are the number of horizontal and vertical viewpoints of the input light field, the maximum value of K is then $n_\rho n_\tau$. $\mathbf{J}_{t,k}$ block calculates the motion tensor of each sub-aperture image ($\boldsymbol{L}(\mathbf{z}, \boldsymbol{\theta}_k)$) at a particular coarse-to-fine level t. The outputs of this block are gradient motion tensor $\left(\mathbf{J}_{t,k}^G\right)$ and intensity motion tensor $\left(\mathbf{J}_{t,k}^g\right)$ which are then summed over all sub-aperture images to form the overall motion tensors $\mathbf{J}_t^G$ and $\mathbf{J}_t^g$ respectively. These motion tensors are then used by Ψ_t' block to calculate the joint motion tensor and robust weights which are required for the iterative solver.

The detailed computation flow of a $\mathbf{J}_{t,k}$ block is presented in Figure 3.3. The input sub-aperture image ($\boldsymbol{L}(\mathbf{z}, \boldsymbol{\theta}_k)$) is downsampled to get a lower resolution (coarser) im-

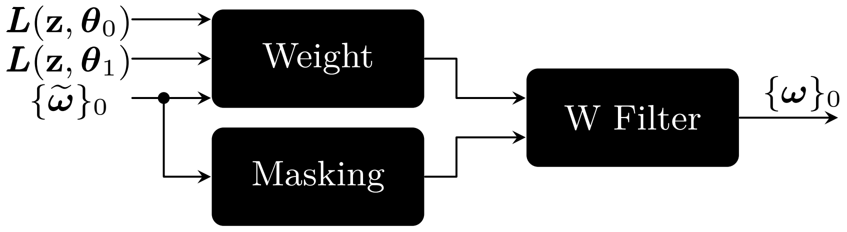


Figure 3.4: Block diagram representation of Weighted-Median-Filtering ($\mathfrak{F}$)

age $\{L(z, \theta_k)\}_t$. A low-pass filter (*'G Filter'*) is applied before downsampling to reduce aliasing. The coarse image $\{L(z, \theta_k)\}_t$ is then shifted toward the coordinate of the reference position θ_0 by previously estimated disparity map ω_t to get a warped image $\{\widehat{L}(z, \theta_k)\}_t$.

For gradient motion tensors, the horizontal and vertical derivatives of warped images need to be calculated. This is done by *Gradient* block as showed in the Figure 3.3. The *Tensor* block calculates the vertical derivative tensor, horizontal derivative tensor and intensity tensor denoted as $\mathbf{J}^{G_x}_{t,k}$, $\mathbf{J}^{G_y}_{t,k}$ and $\mathbf{J}^{g}_{t,k}$ respectively. The gradient motion tensor is then calculated as the average of the two derivative tensors as in Eq. 3.1.

$$\mathbf{J}^{G}_{t,k} = \frac{\mathbf{J}^{G_x}_{t,k} + \mathbf{J}^{G_y}_{t,k}}{2} \tag{3.1}$$

The functionality of *Tensor* unit is the same for all of its three instances. Given a warped sub-aperture image $\tilde{L}(z, \theta_k)$, its motion tensor is calculated as follows

$$\mathbf{J}_k = \begin{bmatrix} \theta_k^{\mathsf{T}} \nabla \tilde{L} \\ \|\theta_k\| \tilde{L}_\theta \end{bmatrix} \begin{bmatrix} \theta_k^{\mathsf{T}} \nabla_2 \tilde{L} & \|\theta_k\| \tilde{L}_\theta \end{bmatrix}. \tag{3.2}$$

With ∇ denotes the spatial gradient operator and $\tilde{L}_\theta$ denotes the directional derivative. These two terms are calculated as follows

$$\begin{aligned} \nabla \tilde{L}(z, \theta_k) &= \begin{bmatrix} \frac{\frac{\partial}{\partial x}\tilde{L}(z,\theta_k) + \frac{\partial}{\partial x}L(z,\theta_0)}{2} \\ \frac{\frac{\partial}{\partial y}\tilde{L}(z,\theta_k) + \frac{\partial}{\partial y}L(z,\theta_0)}{2} \end{bmatrix}, \\ L_\theta(z, \theta_k) &= \frac{\tilde{L}(z, \theta_k) - L(z, \theta_0)}{\|\theta_k\|}. \end{aligned} \tag{3.3}$$

From Eq. 3.3, it is obvious that the reference sub-aperture image $\{L(z, \theta_0)\}_t$ and its derivative images is required for the calculation of motion tensors at each $\mathbf{J}_{t,k}$ block. Instead

Table 3.1: Global work-group size of implemented kernels

#	Kernel	Horizontal size	Vertical size
1	Downscale		
2	G Filter		
3	Gradient	$\left(n_\rho n_x \zeta^t\right)_{\kappa_x}$	$\left(n_\tau n_y \zeta^t\right)_{\kappa_y}$
4	Warp		
5	Tensor		
6	M Filter		
7	Solver	$\left(n_x \zeta^t\right)_{\kappa_x}$	$\left(n_y \zeta^t\right)_{\kappa_y}$
8	Sum $\left(\bigoplus\right)$		
9	Robust weight $\left(\Psi_t'\right)$		
10	Upscale	$\left(n_x \zeta^{t-1}\right)_{\kappa_x}$	$\left(n_y \zeta^{t-1}\right)_{\kappa_y}$
11	Weight		
12	Masking	$\left(n_x\right)_{\kappa_x}$	$\left(n_y\right)_{\kappa_y}$
13	W Filter		

of recalculating this data every time, it is separately calculated and reused for all of $\mathbf{J}_{t,k}$ blocks. This calculation is depicted in gray color in Figure 3.3.

The weighted median filtering computation consists of three units as depicted in Figure 3.4. The *Weight* unit computes the filter weights as described in Eq. 2.44. Beside the reference sub-aperture image $\boldsymbol{L}(\mathbf{z}, \boldsymbol{\theta}_0)$, a neighbor sub-aperture image $\boldsymbol{L}(\mathbf{z}, \boldsymbol{\theta}_1)$, $\boldsymbol{\theta}_1 = [1\ 1]^\top$ and disparity map $\{\widetilde{\boldsymbol{\omega}}\}_0$ are also needed to calculate occlusion weights in Eq. 2.45. The *Masking* unit produces a binary mask indicating the region of interest where weighted median filtering (‘*W Filter*’ unit) should be applied. The generation of this mask includes two steps: detecting the edge in the disparity map using Sobel filter and dilating the edge region by a radius of r_d.

3.2.1 Local and Global Work-group Size

For each kernel execution, a global (work-group) size and a local (work-group) size need to be decided. The global size specifies the number of work-groups which need to be scheduled and the local size specifies the number of work-items associated with one work-group. They are typically defined in the form of a 2-tuple parameter consisting of horizontal and vertical size. The global size needs to be a multiple of the local size. In this work, we apply a common practice that divides the computation work-load by the number of pixel locations in the output. In other words, each work-item will be in charge of calculating the output value for one particular pixel location. Let $\kappa = (\kappa_x, \kappa_y)$ is a local work-group size, where κ_x and κ_y denote the number of work-items in x and y direction respectively. Therefore, each work-group contains $\kappa_x\kappa_y$ work-items and computes output value for a patch of size $\kappa_x \times \kappa_y$. A global size can be simply derived from the local size and the resolution of the output image. Suppose n_x and n_y are respectively the width and the height of the 2D output, global size will be set to $\left((n_x)_{\kappa_x}, \left(n_y\right)_{\kappa_y} \right)$, where operator $(\bullet)_\kappa$ is defined as

$$(n)_\kappa = \kappa\left\lceil \frac{n}{\kappa} \right\rceil, \quad \kappa \in \mathbb{N}, \quad n \in \mathbb{R}. \tag{3.4}$$

Table 3.1 reports the global work-group size of kernels that are implemented in our proposed architecture. The first ten kernels are executed at each coarse-to-fine level in which the resolution of images is changed. Therefore, their sizes are adapted by the scaling factor of ζ and level index t. The last 3 kernels are for the weighted median filter which is executed only one time at the finest level. n_x, n_y, n_ρ, n_τ are the resolution of the input 4D light field. n_x, n_y are the width and the height of a sub-aperture image. n_ρ, n_τ are the number of horizontal and vertical viewpoints respectively. For the first five kernels which are executed on 4D data, the global work-group size is defined to cover a tiling version of all sub-aperture images. The horizontal size and vertical size are then the multiplication of the height and the width of resampled sub-aperture images $(n_y\zeta^t, n_x\zeta^t)$ with the directional resolution (n_τ, n_ρ).

3.2.2 Global Memory Layout

In OpenCL, there are two main approaches to organize data on global memory. The most straightforward approach is using Buffer object in which multi-dimensional data is flattened into a 1D array of bytes. A Buffer object can be addressed using a pointer and is allowed to read and write during kernel execution. Another approach is using Image2D object which offers many advantages over a Buffer object, i.e. texture cache, vector access, auto handling bordering, and pixel interpolation. Texture cache is optimized for 2D spatial locality, so reading image addresses that are close to each other will achieve the best performance [90]. A pixel location must be accessed by a function that will return a vector of color channels (RGBA). The only drawback of Image2D is its memory protection procedure which allows the object to be either write-only or read-only and cannot be both read and write during kernel execution. This limitation indeed has no impact in our case, since the computational flow is arranged for eliminating any read-write requirement on the same memory object.

The access pattern in LF image processing is generally not sequential or structured as normally applied in conventional 2D image processing applications. Depending on the way 4D pixel data is stored on the global memory, it may generate short-burst memory accesses when pixels from a sub-aperture image or an angular patch are read or written [43]. In order to study the impact of memory access on the performance of our accelerator, we investigate two memory layouts for mapping 4D LF image data to an Image2D object: angular major (AM) and spatial major (SM). In SM layout, sub-aperture images are tiled to form a 2D array, while angular patches are tilted in AM layout. Suppose $\mathbf{z}_s, \mathbf{z}_a \in \mathbb{N}^2$ are indices of pixels in SM and AM layout respectively. The relationship between these indices and the coordinates of 4D LF image is expressed in (3.5).

$$\mathbf{z}_s = \begin{bmatrix} n_x & 0 \\ 0 & n_y \end{bmatrix} \boldsymbol{\theta} + \mathbf{z}, \quad \mathbf{z}_a = \begin{bmatrix} n_\rho & 0 \\ 0 & n_\tau \end{bmatrix} \mathbf{z} + \boldsymbol{\theta} \tag{3.5}$$

3.3 Evaluation results

3.3.1 Experimental Setup

We carry out our experiments on both CPU and GPU platforms. A CPU platform equipped with Intel Core i7-5820K (with 6×32 KB L1 cache, 6×256 KB L2 cache, and 15 MB of L3 cache) serves as a baseline implementation. The accelerated architecture is implemented with the OpenCL 1.2 framework and executed on four GPU devices: two Nvidia GPUs (GeForce GTX 1080 Ti, Geforce GTX 680) and two AMD GPUs (Radeon R9 390 and FirePro W9100). Except that Geforce GTX 680 is connected to the PCIe v1 x16 bus, the others are connected to the PCIe v3 x16 bus. Throughout our experiments, GeForce GT 1080 Ti is employed as a default executed device unless otherwise stated. We select this device since it provides the best performance and also is comparable to the related approaches.

Our experiment consists of two parts. In the first part, we quantitatively compare the performance of the proposed approach with state-of-the-art approaches. Both estimation error of the disparity map and processing time are considered. In the second part, a detailed analysis of our accelerated architecture is performed. The runtime on different platforms and the contribution of different parts of the algorithm to the output quality and runtime

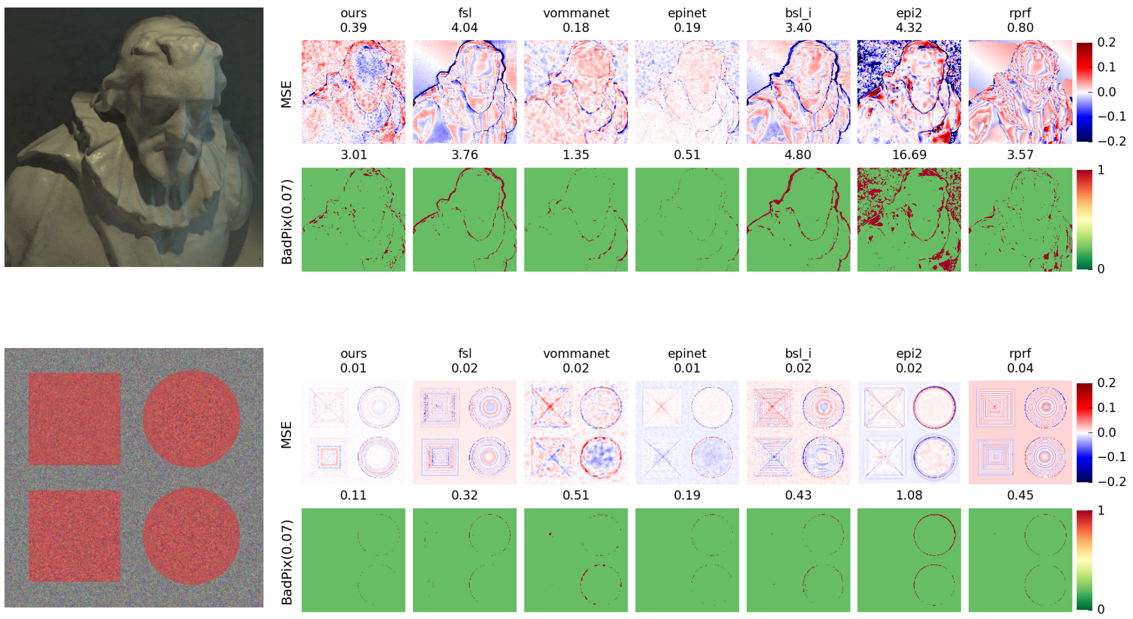


Figure 3.5: Visualization of disparity map quality computed by the fastest seven approaches. Top row and bottom row show the results of *cotton* scene and *pyramids* scene respectively. Left column shows central sub-aperture images and right columns visualize the two error metrics MSE100 and BadPix007.

are reported. Regarding algorithm parameters, we empirically fixed the number of warping levels, scaling factor, local work-group size, and number of solver iteration to 10, 0.85, 16×16, and 100 respectively. Other parameters are data-dependent and are tuned for minimizing disparity error.

3.3.2 Quantitative Comparison

For quantitative evaluation, we employed the 4D light field dataset from Honauer et al. [91]. The dataset contains 25 synthetic LF scenes which are categorized into 4 subsets: *Stratified*, *Training*, *Test* and *Additional*. The first two subsets are selected in this experiment since they include ground-truth disparity maps and results from state-of-the-art algorithms [36]. *Stratified* subset is designed to pose isolated challenges with spatially increasing difficulty, i.e. fine gaps, low textures, noisy regions, and *Training* subset is for simulating photo-realistic scenes. Each LF scene has a resolution of 9×9×512×512 in which 9×9 denotes the directional resolution (81 viewpoints in total) and 512×512 denotes the spatial resolution (or the resolution of each sub-aperture image). The output disparity maps are processed

Table 3.2: Timing and accuracy comparison on Light field Benchmark [36]

Approaches	training								stratified								Avg.	
	dino		cotton		sideboard		boxes		pyramids		stripes		dots		backgammon			
	error	time	error	time	error	time	error	time	error	time	error	time	error	time	error	time	error	time
GVLD	0.407	-0.84	0.395	-1.01	1.241	-0.77	7.159	-1.03	0.005	-0.75	0.880	-0.66	2.096	-1.10	6.329	-0.90	2.314	-0.86
fsl[93]	1.002	0.23	4.039	0.17	3.061	0.40	9.691	0.32	0.019	0.09	3.796	-0.06	9.901	0.13	4.086	0.05	4.449	0.19
vommanet[39]	0.168	0.28	0.184	0.28	0.608	0.28	1.924	0.27	0.024	0.27	0.914	0.27	1.154	0.27	3.304	0.28	1.035	0.28
epinet[25]	0.167	0.29	0.191	0.29	0.827	0.29	6.240	0.30	0.008	0.29	0.950	0.29	1.635	0.29	3.629	0.30	1.706	0.29
bsl_i[33]	1.088	0.57	3.395	0.61	8.907	1.06	11.304	0.89	0.024	0.27	5.427	0.41	14.014	0.04	5.386	0.25	6.193	0.63
epi2[88]	2.076	0.94	4.318	0.96	4.651	0.94	10.928	0.95	0.022	0.84	6.104	0.93	6.657	0.88	20.748	0.84	6.938	0.91
rprf[94]	0.615	1.16	0.803	1.05	1.241	1.14	10.651	1.11	0.043	1.08	9.112	0.92	20.832	1.04	4.948	1.01	6.031	1.07
epi_shift[40]	0.392	1.27	0.475	1.28	1.261	1.28	9.790	1.28	0.037	1.54	1.686	1.44	13.154	1.44	12.788	1.44	4.948	1.38
rm3de[31]	0.360	1.65	0.341	1.66	1.071	1.65	7.625	1.75	0.058	1.66	1.001	1.54	3.293	1.54	9.212	1.67	2.870	1.65
lfvar[72]	0.482	1.70	0.468	1.58	1.343	1.69	7.179	1.72	0.007	2.13	0.920	1.53	2.121	1.39	6.728	1.43	2.406	1.71
soa_epn[37]	0.471	1.86	0.484	1.87	1.167	1.88	7.698	1.86	0.020	1.86	1.296	1.84	24.004	1.85	4.016	1.86	4.895	1.86
epi1[95]	1.226	1.93	2.247	1.93	2.852	1.93	8.717	1.93	0.027	1.96	2.670	1.96	5.730	1.91	9.559	1.93	4.128	1.94
ofsy[35]	0.782	2.30	2.653	2.27	2.478	2.34	9.561	2.47	0.008	2.29	7.269	2.09	14.756	1.99	7.549	2.41	5.632	2.29
fbs3[34]	0.662	2.83	0.764	2.78	1.158	2.82	8.904	2.88	0.029	2.85	1.315	2.62	2.088	2.72	5.669	2.68	2.574	2.78
cae[23]	0.382	2.92	1.506	2.91	0.876	2.94	8.424	2.92	0.048	2.93	3.556	2.88	5.082	2.98	6.074	2.91	3.243	2.92
lf[28]	1.164	3.05	9.168	2.99	5.071	2.99	17.434	2.98	0.273	2.97	17.454	3.04	5.676	2.99	13.007	2.99	8.656	3.00
ps_rf[96]	0.751	3.11	1.161	3.09	1.945	3.01	9.043	3.08	0.043	2.97	1.382	3.04	8.338	2.99	6.892	2.99	3.694	3.04
sc_gc[32]	0.504	3.28	1.865	3.28	1.823	3.36	12.286	3.29	0.072	3.32	15.622	3.34	19.812	3.36	5.647	3.26	7.204	3.31
spo[97]	0.310	3.31	1.313	3.31	1.024	3.32	9.107	3.33	0.043	3.35	6.955	3.29	5.238	3.33	4.587	3.34	3.572	3.32
lf_occ[30]	1.137	4.00	1.068	3.80	2.304	4.13	9.850	4.02	0.098	4.07	8.131	4.29	3.301	4.03	21.587	3.71	5.934	4.04

(red: best, green: second best, blue: third best)

using the HCI evaluation toolkit [92] and compared to benchmark results [36] published by the end of 2019 when the proposed approach was published.

Table 3.2 reports the accuracy (*error*) and computation time (*time*) of state-of-the-art approaches on 8 LF scenes and their average. *error* (MSE100) is the mean square error of the estimated disparity map multiplying by 100 and *time* is log_{10} of runtime in second [1]. The proposed approach (GVLD) achieves the fastest runtime on all LF scenes and shows the third-best accuracy on average. In particular, we score the best accuracy on two *stratified* LF scenes (*pyramids* and *stripes*) and the third-best accuracy on one *training* LF scene (*boxes*). This high accuracy (i.e. *pyramids* scene) pays tribute to the intrinsic sub-pixel precision of variational principle which models disparity map as a continuous function. We notice that the main source of error in our estimated disparity map comes from the occluded and reflected regions where the photo constancy assumption becomes less reliable. Compared to our previous work lfvar [72], GVLD demonstrates better performance in both accuracy and processing speed. For all test scenes, GVLD provides lower errors and more than 350× faster runtime. The improvement in accuracy is due to the application of WMF in which occlusion information is taken into account. The two approaches that achieve the best accuracy are epinet [25] and vommanet [39] which are based on deep learning techniques. This precision comes with the cost of a large amount of data and time for training end-to-end convolutional networks. Our approach, in contrast, works

[1]We follow the standard metrics provided with the benchmark [91].

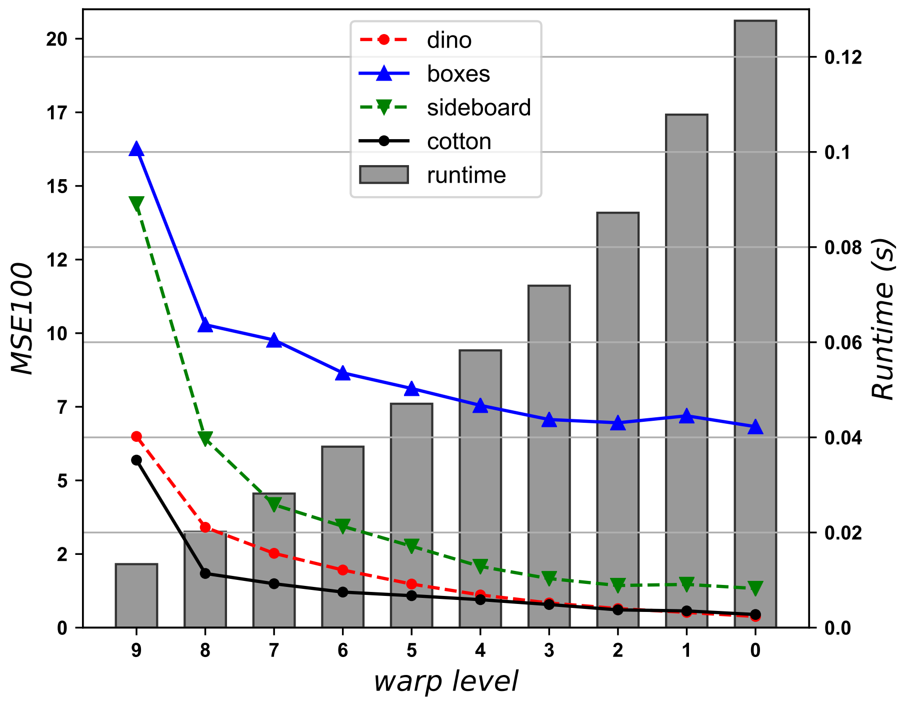


Figure 3.6: Runtime and accuracy at different warping levels.

independently from training data and can be directly deployed for new LF data. In addition, our approach outperforms these deep learning approaches in the term of processing time considering the same execution platform (Geforce 1080 TI). The difference in runtime is more than 10× on average. This is due to the high complexity of their convolutional networks for mimicking the disparity estimation process. It is computationally expensive and leads to long inference time even running on optimized deep learning frameworks such as Tensor Flow.

Figure 3.5 visualizes the quality of disparity maps computed by the fastest seven approaches. Besides the *MSE100* metric, we also report the *BadPix007* metric which denotes the percentage of pixels whose absolute error compared to the ground truth is larger than 0.07. The proposed approach presents an extremely low pixel error ratio (0.11%) compared to the other approaches on *pyramids* scene and score the third position in *cotton* scene.

3.3.3 Performance Analysis

As discussed earlier, our approach employed the coarse-to-fine warping strategy to cope with the large displacement issue. This strategy gradually refines disparity maps in each warping level and produces the final output with higher precision. This accuracy indeed comes with more computation effort as shown in Figure 3.6. The plot demonstrates the impact of this strategy on accuracy and runtime at different warping levels. The number of warp levels is set to 10 with the scaling factor $\zeta = 0.85$. The spatial resolutions at level 0 (finest level) and level 9 (coarsest level) are 512×512 and 118×118 respectively. At the end of each warping level, the estimated disparity map is upsampled to the original resolution and compared to the ground-truth. We report the error (*MSE100*) and the average of accumulated runtime (in second) of four LF scenes in *training* subset. In this test, we execute our approach on GeForce 1080 TI using the same algorithm parameters as in the previous section. It could be seen from Figure 3.6 that the results of the coarsest level show large errors compared to the ground-truth due to the lack of fine structure information. However, this error is gradually reduced after each warping level and reaches the minimum at the finest resolution. While the accumulated runtime steadily increases over 10 warping phases, the improvement in accuracy is sharp for the first 7 levels and slows down in the last 3 levels. The reason for this behavior is that the smaller the warping level is, the less newly fine structure information is added to the previous level. The improvement in accuracy is, therefore, smaller when compared to coarser levels. In addition, for the scene which consists of large occluded area (i.e. boxes), the disparity error at these areas can affect the overall error and leads to a fluctuation in the last three levels as can be seen in Figure 3.6. This observation suggests a way to further achieve a faster processing speed with a small loss of accuracy.

To understand the impacts of different parts of the proposed approach to the accuracy, we experimented on a challenging light field scene (*boxes*), see Figure 3.7. In this experiment, four configurations of the proposed algorithm are tested and compared to a multi-label optimization approach (*lf* [28]). *lf* was selected for this evaluation, since it shares many similarities to ours in approaching the disparity estimation problem. Two constancy assumptions including photo constancy and gradient constancy are employed to characterize disparity map and guided median filtering is applied to refine the output. The main difference between *lf* and ours is that *lf* bases on multi-label optimization while our approach bases on the variational principle. A multi-label optimization approach requires a discretization of disparity value to create a set of labels that are then assigned to each pixel in the disparity map. Since the disparity range is unknown, this type of approach generally struggles with the loss of accuracy due to discretization and the trade-off between the number of labels and the computation requirements. As could be seen in Figure 3.7 (c), the effect of discretized disparity labels can be observed in *lf*'s result. Our approach, on the other hand, benefits from the intrinsic sub-pixel precision of variational formulation and provides a smooth and accurate result. Even without post-processing, the MSE100 error of

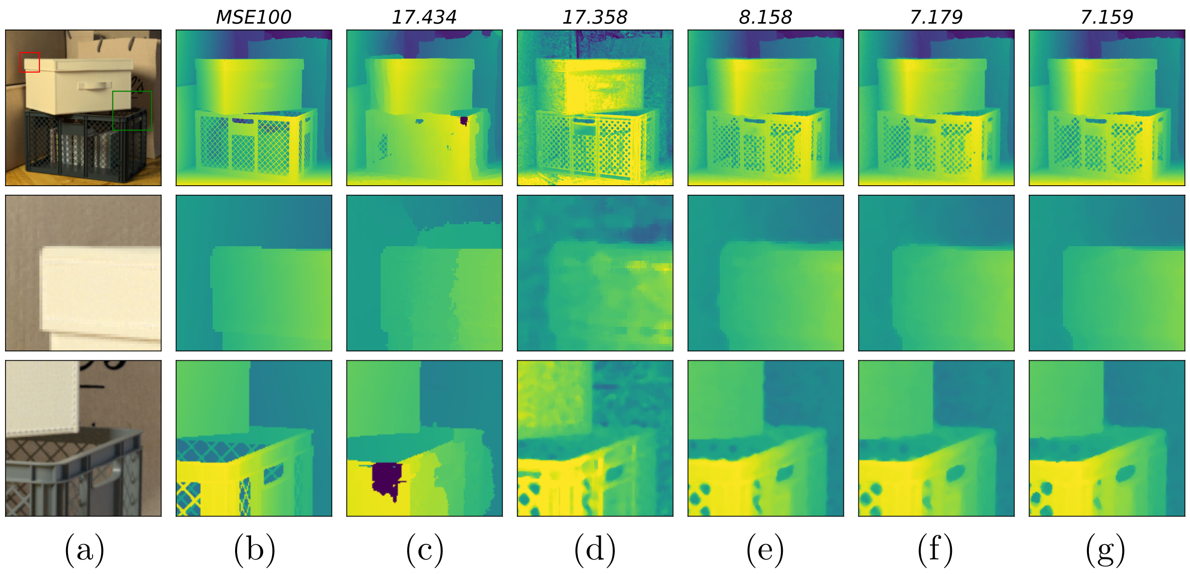
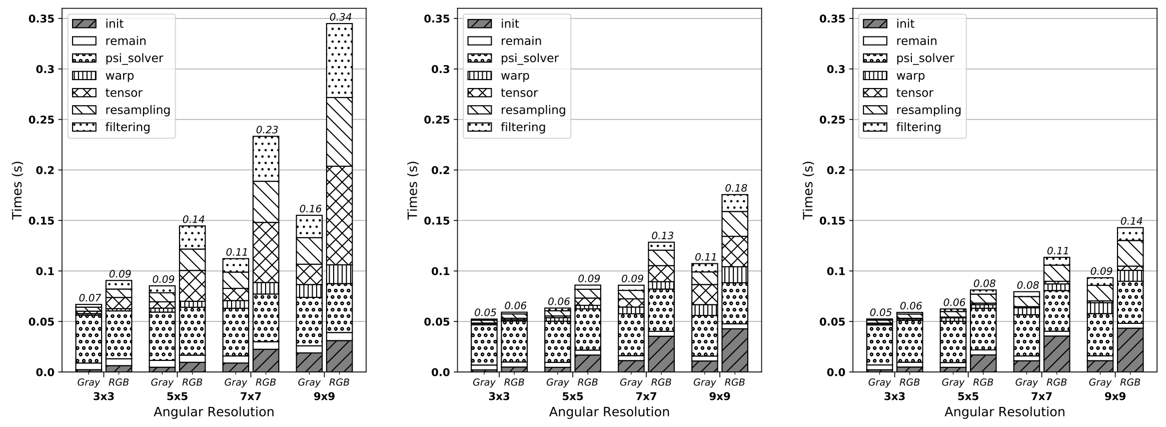


Figure 3.7: Comparison of disparity maps estimated by different approaches and configurations on light field scene “*boxes*”. The full-size sub-aperture image and disparity maps are presented in the first row, while two zoom-in regions marked with red and green rectangles are in the second and the third row respectively. (a) center sub-aperture image; (b) ground-truth disparity map; (c) multi-label optimization approach (lf [28]); (d) ours, without warping strategy and post-processing; (e) ours, without post-processing; (f) ours, post-processing without occlusion weights (lfvar [72]); (g) ours, post-processing with occlusion weights.

our approach is less than half of *lf*, Figure 3.7 (e). It is also noticed that the warping strategy plays an important role in the proposed algorithm. Without warping, the linearization of (2.5) becomes invalid and leads to wrong estimation of the disparity values, as in Figure 3.7 (d). The contribution of post-processing to the accuracy is also evident in Figure 3.7 (f) and (g). Post-processing corrects the disparity value at disparity edge area and leads to a reduction of 1 in MSE100 error metric. Compared to our previous work (lfvar [72]), the use of WMF which includes occlusion weights provides a slight improvement in the accuracy.

Each part of the proposed approach contributes differently to the overall processing time. For a deeper analysis of this timing concern, the implemented program is divided into seven main parts for which timing profiling is conducted with various programming configurations. Besides the number warping level, scaling factor, and the number of iterations that are fixed, the other parameters that affect the running time are color schemes (grayscale, RGB) and angular resolution (3×3, 5×5, 7×7, 9×9). In addition to these parameters, we also analyze the impacts of global memory layouts, discussed in Section 3.2.2. The timing profiling is



(a) Buffer object (b) Image2D: AM layout (c) Image2D: SM layout

Figure 3.8: Timing profiling of main processes running at different configurations. *init*: memory transfer and variable initialization. *filtering*: kernels 2,6. *resampling*: kernels 1,10. *warp*: kernel 4. *tensor*: kernels 3,5,8. *solver*: kernels 7-9. *remain*: function calls, argument processing.

done by executing the program 100 times for each setup and reporting the average runtime. The experimental results are shown in Figure 3.8. Overall, Image2D object demonstrates better performance compared to Buffer object. This pays tribute to the texture cache running behind every access to pixel data. Using Image2D object, the SM layout provides a higher performance compared to AM layout, especially for higher angular resolutions. The most noticeable improvement is come from the *tensor* part of our accelerator. This behavior is expectable because LF motion tensor heavily relies on the calculation of derivatives. The amount of memory access for spatial derivative dominates the memory access for angular derivative in the proposed approach. *psi_solver* computes the robust weights and iterative solutions of the Jacobian solver. The input to this part, joint light field motion tensor, has the same structure regardless of programming configuration. It leads to a constant running time as can be seen in Figure 3.8. The runtime of *init* part is proportional to angular data size since the most time-consuming task during the initialization is transferring LF data from the host computer to GPU global memory.

In the proposed approach, WMF is applied to the output of the last warping level to enhance its quality. This step indeed improves the accuracy of the estimated disparity map. However, we notice that its contribution is only significant in the regions around the depth discontinue where most of the erroneous pixels are located. Instead of running this filter on the whole image, we generate a binary mask identifying pixels with a high potential of error and use it as a guide to applying WMF. Figure 3.9 demonstrates use of WMF on the disparity map of *dino* scene. The blurry regions around the depth edges, Figure 3.9 (f), (g), are corrected by WMF and become sharper and more accurate, Figure 3.9 (h), (i).

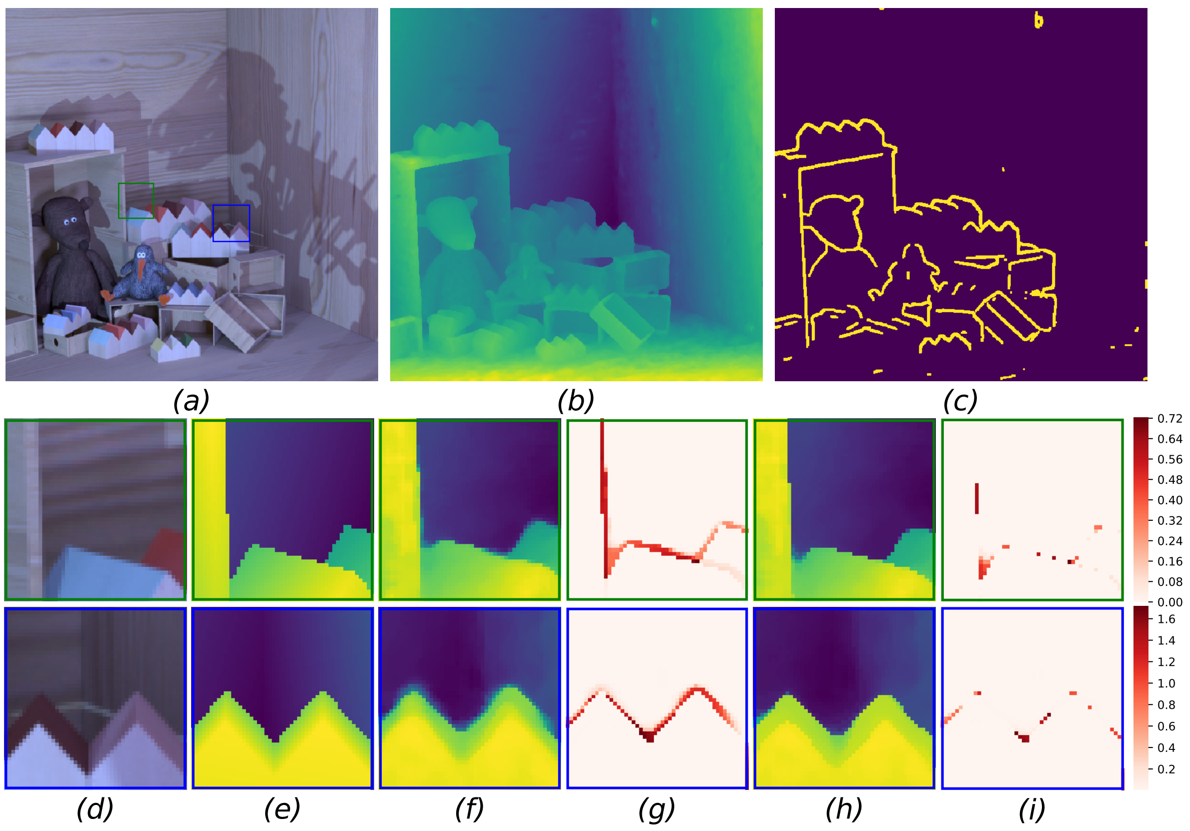


Figure 3.9: Improvement of accuracy when applying WMF on *dino* scene: (a) sub-aperture image with two regions of interest (ROI); (b) estimated disparity map; (c) binary map of selected region; (d) zoom-in ROIs; (e) ground-truth; (f), (h) disparity map before and after WMF respectively; (g), (i) MSE of (f) and (g) respectively;

Figure 3.10 reports the runtime of WMF with five different window sizes increasing from 3×3 to 11×11. There are two strategies are tested, applying to all the pixels (*full*) and applying to selected pixels (*selective*) using the binary mask in Figure 3.9 (c). Compared to the first strategy the second strategy requires less computation effort and runtime. We can achieve the speed-up of 5× when using *selective* strategy in the case of large window size (i.e. 11×11).

Figure 3.11 shows the average runtime of the proposed approach running on 4 GPU platforms. We report the runtime of three memory layouts and compare them to a CPU implementation. CPU runtime is 47.3s which is 80-255× slower than our *buffer* object version and 128-365× slower than the version using Image2D object (SM Layout).

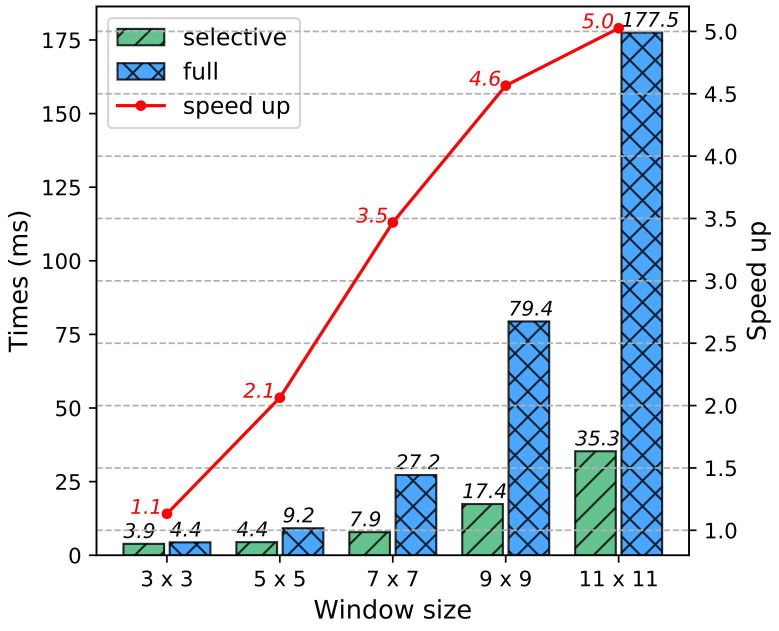


Figure 3.10: Runtime evaluation of weighted median filtering operator (kernels 11-13). Selective and full strategy are applied to *dino* scene.

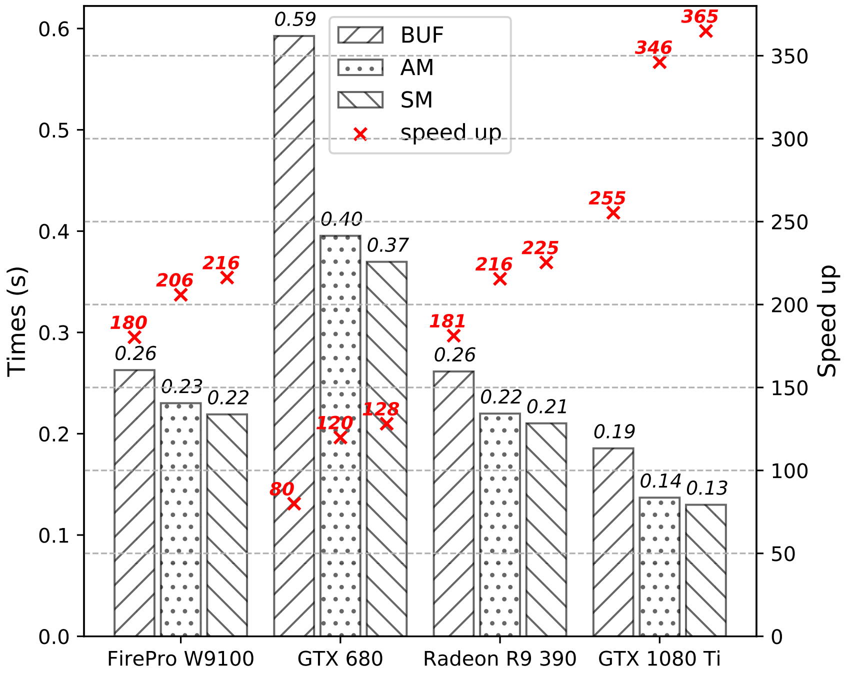


Figure 3.11: Average execution time of the proposed approach on different platforms.

3.4 Summary

In this chapter, a GPU accelerated approach for light field disparity estimation is presented. The numerical computation to iteratively compute the solution derived in the previous chapter is effectively realized under OpenCL framework. In order to evaluate the performance of the proposed approach considering both accuracy and timing aspects, an extensive evaluation is conducted on synthetic 4D LF dataset. The quantitative comparison shows our superior performance compared to the previous approaches. On average, the proposed approach is 10+ times faster than published works running on a similar GPU platform and provides the most accurate solution among optimization-based approaches. Experiment on several GPU platforms (Nvidia and AMD GPUs) show a speed up more than $80\times$ as compared to the CPU implementation.

CHAPTER 4

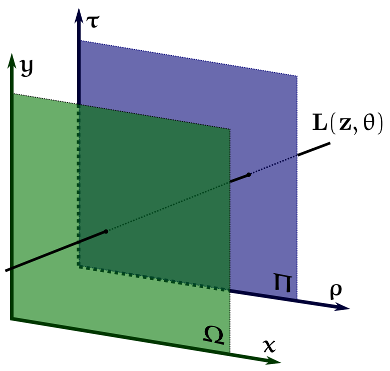
GPU-ACCELERATED 4D LIGHT FIELD SUPER-RESOLUTION

This chapter discusses a GPU accelerated approach for the reconstruction of spatially high-resolution perspective image from low-resolution 4DLF degraded by mixed Gaussian-Impulse noise. The LF super-resolution model derived from the statistical perspective consists of a joint ℓ^1-ℓ^2 data fidelity term and a weighted regularization term. While the first term provides a proper treatment to mixed Gaussian-Impulse noise conditions, the second term introduces an effective way to integrate image features for a better regularization effect. The proposed optimization problem can be effectively solved with the alternating direction method of multipliers (ADMM). A GPU accelerated architecture is presented for speeding up the iterative solver. Experiments on synthetic 4DLF dataset [91] and natural image dataset DIV8K [98] are conducted to evaluate the performance of the proposed approach and compared to related works. Parts of the results of this chapter have been published in [48, 99].

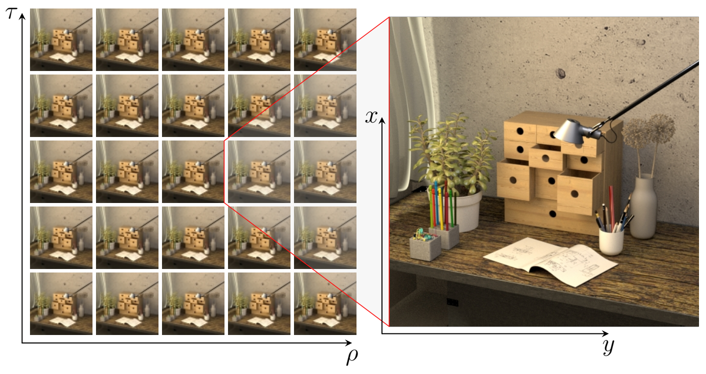
4.1 Degradation Model and Notation

Light field is a 4D parameterization of the plenoptic function [2], it is visually described as a ray indexed by its intersection with two parallel planes, as depicted in Figure 4.1(a).

$$\boldsymbol{L}\colon \Omega\times\Pi \to \mathbb{R}, \qquad (\mathbf{z}, \boldsymbol{\theta}) \to \boldsymbol{L}(\mathbf{z}, \boldsymbol{\theta}) \tag{4.1}$$



(a)



(b)

Figure 4.1: Light field representation and acquisition; (a) Two-plane parameterisation; (b) 2D array representation of sub-aperture images (SAIs).

where $\mathbf{z} = [x, y]^T$ and $\boldsymbol{\theta} = [\rho, \tau]^T$ denote coordinate pairs in the spatial plane $\Omega \subset \mathbb{R}^2$ and in the directional plane $\Pi \subset \mathbb{R}^2$ respectively. In practice, the value of this function can be a vector of 3 color components (i.e., RGB color light field) and the two planes are discretized by the sampling rate of capturing devices (e.g., sensor size, number of cameras,...). Given a 4D light field $\boldsymbol{L}$, spatial information could be obtained from one perspective by keeping the directional component $\boldsymbol{\theta}$ unchanged and varying over spatial domain Ω. Such spatial information gives us a sub-aperture image (SAI), or a view, of the captured scene. Fig. 4.1(b) shows a 5×5 angular views of LF scene '*table*' [91]. From this perspective, a 4D LF is a collection of 2D images captured from different viewpoints and the reconstruction of high-resolution SAIs shows a strong connection to the multi-image super-resolution (MISR) problem.

Let us rearrange the 2D angular view of low-resolution SAIs into a 1D set of s_k low-resolution observations $Y_k \in \mathbb{R}^{s_y \times s_x}$, $k \in [1, s_k]$. Our goal is to approximate the high-resolution version $X \in \mathbb{R}^{s_Y \times s_X}$, where $s_y \times s_x$ and $s_Y \times s_X$ are the size of low-resolution images and the size of high-resolution image, respectively. In practice, a low-resolution image Y_k is considered as a degraded version of the high-resolution image X. This degradation can be modelled by the application of three linear operators: warping ($\mathcal{W}_k$), blurring ($\mathcal{B}$), and down-sampling ($\mathcal{D}$), as depicted in Fig. 4.2. The warping operator represents the positioning of the camera. Shifting the camera's position will result in the corresponding shifts of pixels in the captured image. This effect is actually the source of the angular dimension in 4D Light field. We define the warping operator as $\mathcal{W}_k : \mathbb{R}^{s_Y \times s_X} \longrightarrow \mathbb{R}^{s_Y \times s_X}$ which transforms a high-resolution image into a new one observed from a different perspective. The blurring operation represents the point spread function (PSF) which describes the response of an imaging system to an object point. Depending on the setup of lenses and

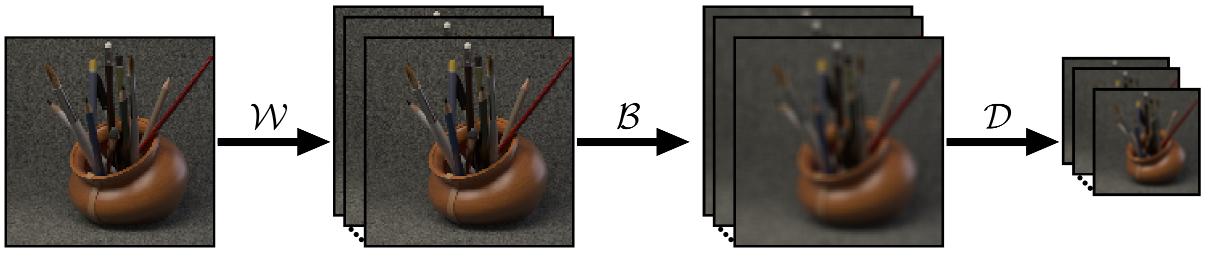


Figure 4.2: Degradation process

imaging sensors, PSFs can become very complicated and even spatially variant. However, as shown in the literature [24, 63, 100, 101], it is sufficient to assume a spatially invariant version of PSF which can be modelled by a linear operator, i.e. $\mathcal{B} : \mathbb{R}^{s_Y \times s_X} \longrightarrow \mathbb{R}^{s_Y \times s_X}$. The down-sampling operator represents the digital sampling process of an imaging sensor, i.e., $\mathcal{D} : \mathbb{R}^{s_Y \times s_X} \longrightarrow \mathbb{R}^{s_y \times s_x}$. As a combination of these linear operators, the image foundation process can be described as

$$Y_k = \mathcal{D}_k \circ \mathcal{B} \circ \mathcal{W}_k(X) + \epsilon_k, \forall k \in [1, s_k], \tag{4.2}$$

where ϵ_k represents the measurement error or the additive noise which is practically assumed to follow the Gaussian distribution or Laplace distribution. For a better presentation, we transform the Eq. 4.2 into its vector form:

$$\mathbf{y}_k = DBW_k\mathbf{x} + \boldsymbol{\epsilon}_k. \tag{4.3}$$

In Eq. 4.3, $\mathbf{y}_k, \boldsymbol{\epsilon}_k \in \mathbb{R}^{s_x s_y}$ and $\mathbf{x} \in \mathbb{R}^{s_X s_Y}$ are the column-vector representations of Y_k, ϵ_k and X. Linear transformation matrices D, B, and W_k respectively replaced the linear operators $\mathcal{D}$, $\mathcal{B}$, and $\mathcal{W}_k$. To further simplify the notation, we define $p = s_X s_Y$, $q = s_x s_y$, and combine D, B and W_k into A_k, i.e., $A_k = DBW_k$. It follows that $B, W_k \in \mathbb{R}^{p \times p}$, $D \in \mathbb{R}^{q \times p}$, and $A_k \in \mathbb{R}^{q \times p}$.

4.2 Bayesian Image Super-resolution Framework

Let us start with the standard Bayesian formulation which poses super-resolution problem as finding a maximum a posteriori (MAP) high-resolution image $\mathbf{x}$ given a set of low-resolution samples $\{\mathbf{y}_k \mid k = 1, .., s_k\}$

$$\mathbf{x} = \arg\max_{\mathbf{x}} \mathcal{P}(\mathbf{x}|\mathbf{y}_i, ..., \mathbf{y}_{s_k}),$$

where $\mathcal{P}(\mathbf{x}|\mathbf{y}_i, ..., \mathbf{y}_{s_k})$ is called posterior and represents the conditional probability density of $\mathbf{x}$ given the set of degraded images $(\mathbf{y}_k)$. Follows Bayes' rule, we have

$$\mathcal{P}(\mathbf{x}|\mathbf{y}_1, ..., \mathbf{y}_{s_k}) = \frac{\mathcal{P}(\mathbf{x}) \prod\limits_{k=1}^{s_k} \mathcal{P}(\mathbf{y}_k|\mathbf{x})}{\prod\limits_{k=1}^{s_k} \mathcal{P}(\mathbf{y}_k)},$$

with $\mathcal{P}(\mathbf{y}_k|\mathbf{x})$ is likelihood function which encodes the likelihood that the high-resolution image $\mathbf{x}$ is due to the low-resolution observation $\mathbf{y}_k$. The likelihood function is defined with an assumption of the known noise model of ϵ_k in the observation of $\mathbf{y}_k$. Here, we assume that the noise affecting the observed low-resolution image $\mathbf{y}_k$ is independent. Therefore, the likelihood $\mathcal{P}(\mathbf{y}_1, ..., \mathbf{y}_{s_k}|\mathbf{x}) = \prod\limits_{k=1}^{s_k} \mathcal{P}(\mathbf{y}_1|\mathbf{x})$ and the normalization factor $\mathcal{P}(\mathbf{y}_1, ..., \mathbf{y}_{s_k}) = \prod\limits_{k=1}^{s_k} \mathcal{P}(\mathbf{y}_k)$. $\mathcal{P}(\mathbf{x})$ is an image prior describing the properties of the high-resolution image being reconstructed. Since the low-resolution samples are known, $\mathcal{P}(\mathbf{y}_k), k = 1, ..., s_k$ are constants, the above MAP problem can be transformed into a minimization of negative log-likelihood

$$\arg\max_{\mathbf{x}} \mathcal{P}(\mathbf{x}|\mathbf{y}_1, ..., \mathbf{y}_{s_k}) = \arg\min_{\mathbf{x}} -ln\mathcal{P}(\mathbf{x}) - \sum_{k=1}^{s_k} ln\mathcal{P}(\mathbf{y}_k|\mathbf{x}).$$

The above two logarithmic terms represent the typical setup of an optimization problem consisting of a data-fidelity term (i.e., $E(\mathbf{x}) := -\sum_{k=1}^{s_k} ln\mathcal{P}(\mathbf{y}_k|\mathbf{x})$) and a regularization term (i.e., $R(\mathbf{x}) := -ln\mathcal{P}(\mathbf{x})$) as shown in Eq. 4.4. In the following sections, the choices of these two terms are discussed in more detail.

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \mathcal{P}(\mathbf{x}|\mathbf{y}_1, ..., \mathbf{y}_{s_k}) := \arg\min_{\mathbf{x}} E(\mathbf{x}) + R(\mathbf{x}) \tag{4.4}$$

4.2.1 The Data-Fidelity Term

The construction of the data fidelity term depends on the noise models which are commonly assumed to follow Gaussian and Laplace distribution [102].

For additive Gaussian noise, $\epsilon_k \sim \mathcal{N}(\mu_k, \sigma_k^2)$ follows a normal distribution with the probability density function given by $\left(1/\sqrt{2\pi\sigma_k^2}\right) e^{-(\mu_k - \epsilon_k)^2/2\sigma_k^2}$. Assuming a 0 central distribution (i.e. $\mu_k = 0$), the likelihood function $\mathcal{P}(\mathbf{y}_k|\mathbf{x})$ reads

$$\mathcal{P}(\mathbf{y}_k|\mathbf{x}) \propto \exp\left(\sum_{k=1}^{s_k} \|A_k\mathbf{x} - \mathbf{y}_k\|_2^2\right),$$

which results in a well-known least square fidelity term. In the case of Laplace noise (i.e., impulse noise), $\epsilon_k \sim \mathcal{L}(\mu_k, b)$ has the probability density function given by $\frac{1}{2b}e^{-\|\mu_k-\epsilon_k\|_1/b}$,

$$\mathcal{P}(\mathbf{y}_k|\mathbf{x}) \propto \exp\left(\sum_{k=1}^{s_k} \|A_k\mathbf{x} - \mathbf{y}_k\|_1\right).$$

This results in an ℓ^1 norm data fidelity term, which shows robustness against outliers and superior performance with impulse noise [63, 103]. In order to handle the mixed Gaussian-impulse noise situation, we followed the previous works [104, 105] to combine ℓ^1 and ℓ^2 norm resulting in a joint $\ell^1 - \ell^2$ data fidelity term,

$$E(\mathbf{x}) = \sum_{l\in\{1,2\}} \lambda_l \sum_{k=1}^{s_k} \|A_k\mathbf{x} - \mathbf{y}_k\|_l^l,$$

with parameters λ_1 and λ_2 control the contribution of ℓ^1 and ℓ^2 norm respectively.

4.2.2 Regularization Term

In Bayersian framework, it is generally assumed that $\mathbf{x}$ is an Markov random field (MRF) with a strictly positive joint probability density. Therefore, following Hammersly-Clifford theorem, its joint probability density must have the form of a Gibbs distribution[106–108]:

$$\mathcal{P}(\mathbf{x}) := \frac{1}{Z}exp\left(-\frac{1}{T}\sum_{C\in C} V_C(\mathbf{x})\right), \tag{4.5}$$

where Z is a normalizing constant, T stands for *temperature* and controls the degree of peaking [108]. V_C is called potential defined for a local group of pixels or clique C. The sum is for a set C of all possible cliques. The definition of clique set C and the selection of the potential V_C lead to various types of image prior, which practically share the following form

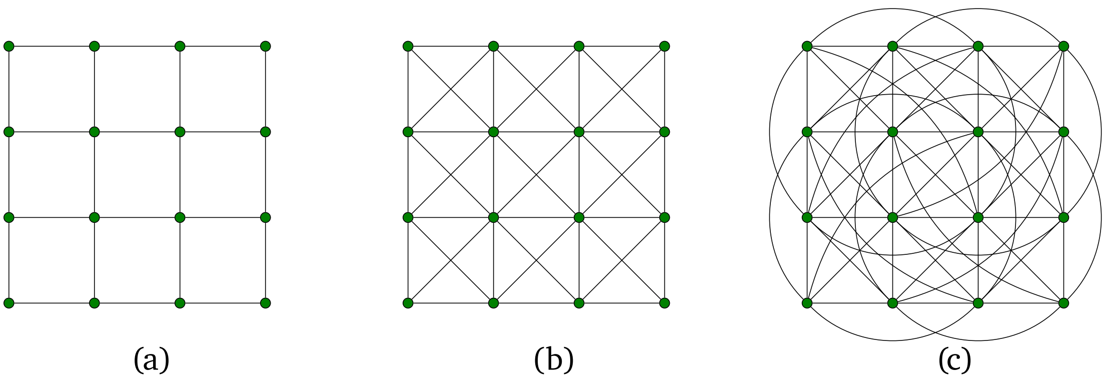


Figure 4.3: Configurations of $\mathcal{N}(\mathbf{u})$: (a) 4 neighbors; (b) 8 neighbors; (c) 16 neighbors.

$$\mathcal{P}(\mathbf{x}) \propto \exp\left(\sum_{\mathbf{u}\in\Omega}\sum_{\mathbf{v}\in\mathcal{N}(\mathbf{u})} w(\mathbf{u},\mathbf{v})\Phi(\mathbf{x}_\mathbf{u},\mathbf{x}_\mathbf{v})\right), \tag{4.6}$$

where $\mathbf{u},\mathbf{v} \in \Omega$, with $\Omega \subset \mathbb{N}^2$ represents the 2D indices of $\mathbf{x}$. $w : \Omega^2 \longrightarrow \mathbb{R}$ and $\phi : \mathbb{R}^2 \longrightarrow \mathbb{R}$ are respectively weighting function and distance function. The weighting function characterizes the dependency in pixel locations, while the distance function penalizes the difference in pixel intensities. $\mathcal{N}(\mathbf{u})$ represents a set of indices defined with regarding to the index $\mathbf{u}$, as in Fig. 4.3. For example, if we set the distance function $\Phi(\mathbf{x}_\mathbf{u},\mathbf{x}_\mathbf{v}) = \mathbf{x}_\mathbf{u}^2 + \mathbf{x}_\mathbf{v}^2$, the weighting function $w(\mathbf{u},\mathbf{v}) = \frac{\alpha}{2}$ and limit $\mathcal{N}$ to a single index (i.e., $\mathcal{N}(\mathbf{u}) = \{\mathbf{u}\}$), Eq. 4.6 gives us the following ℓ^2 regularization term

$$R(\mathbf{x}) = \alpha\, \|\mathbf{x}\|_2^2 = \alpha \sum_{\mathbf{u}\in\Omega} \mathbf{x}_\mathbf{u}^2 \tag{4.7}$$

In another setup, let us set the distance function $\Phi(\mathbf{x}_\mathbf{u},\mathbf{x}_\mathbf{v}) = (\mathbf{x}_\mathbf{u} - \mathbf{x}_\mathbf{v})^2$ and the weighting function $w(\mathbf{u},\mathbf{v}) = \alpha$. $\mathcal{N}(\mathbf{u})$ has two indices, one is immediately to the right of $\mathbf{u}$ (denoted as $\mathbf{u}_{x+1}$) and the other is immediately below of $\mathbf{u}$ (denoted as $\mathbf{u}_{y+1}$). Eq. 4.6 leads to another ℓ^2 regularization on the gradient of $\mathbf{x}$.

$$R(\mathbf{x}) = \alpha\, \|\nabla\mathbf{x}\|_2^2 = \alpha \sum_{\mathbf{u}\in\Omega} (\mathbf{x}_\mathbf{u} - \mathbf{x}_{\mathbf{u}_{x+1}})^2 + (\mathbf{x}_\mathbf{u} - \mathbf{x}_{\mathbf{u}_{y+1}})^2 \tag{4.8}$$

The regularization terms in Eq. 4.7 and Eq. 4.8 belong to the Tikhonov regularization family

$$R(\mathbf{x}) = \|\Gamma\mathbf{x}\|_2^2,$$

where Γ indicates the Tikhonov matrix which can be chosen as an identity matrix (Eq. 4.7), preferring the solutions with smaller norm, or a difference operator (Eq. 4.8), enforcing the smoothness of the solution.

The Tikhonov regularization is a practical choice to improve the conditioning of the problem while keeping the computation effort not too high. However, a common drawback of this approach is over-smoothing problem leading to poor performance on images with many discontinues (e.g., dense texture regions, occlusions,...). A well-known alternative to Tikhonov regularization is total variation (TV) regularization [109]

$$R(\mathbf{x}) = \|\nabla \mathbf{x}\|_1 .$$

This prior shares a similar setup as Eq. 4.8, except that the distance function is set to an absolute difference, i.e., $\Phi(\mathbf{x}_i, \mathbf{x}_k) = |\mathbf{x}_i - \mathbf{x}_k|$. Compared to ℓ^2 norm, the ℓ^1 norm performs better on edges where there is a rapid change in pixels' values. The piece-wise smooth effect of TV prior is generally desirable and contributes to the quality of the final solution. As a combination of TV prior and bilateral filter, Farsiu et al. introduce Bilateral TV (BTV) regular [63]

$$R(\mathbf{x}) = \alpha \sum_{\mathbf{u}\in\Omega} \sum_{\mathbf{v}\in\mathcal{N}_w(\mathbf{u})} \lambda^{\|\mathbf{u}-\mathbf{v}\|_1} |\mathbf{x}_\mathbf{u} - \mathbf{x}_\mathbf{v}|, \tag{4.9}$$

where $\mathcal{N}_w(\mathbf{u})$ contains the indices the lie in the lower triangular of a square window centered at $\mathbf{u}$. In Eq. 4.9, the weighting function has the form of $w(\mathbf{u},\mathbf{v}) = \lambda^{\|\mathbf{u}-\mathbf{v}\|_1}$, with $0 < \lambda < 1$, giving a spatially decaying effect to the summation of the regularization terms. The BTV prior can be considered as a windowed version of Non-local Total Variation [110, 111] (NLTV) defined in the form

$$\mathrm{R}(\mathbf{x}) = \sum_{\mathbf{u}\in\Omega} \sum_{\mathbf{v}\in\Omega} \frac{|\mathbf{x}_\mathbf{u} - \mathbf{x}_\mathbf{v}|}{\tilde{d}(\mathbf{u},\mathbf{v})}, \tag{4.10}$$

where $0 < \tilde{d}(\mathbf{u},\mathbf{v}) \leq \infty$ is a positive measure defined between the points $\mathbf{u}$ and $\mathbf{v}$.

In this work we focus on the following weighted regularization term,

$$\mathrm{R}(\mathbf{x}) = \sum_{\mathbf{u}\in\Omega} \sum_{\mathbf{v}\in\mathcal{N}(\mathbf{u})} w(\mathbf{u},\mathbf{v}) |\mathbf{x}_\mathbf{u} - \mathbf{x}_\mathbf{v}|, \tag{4.11}$$

which can be considered as a generalized version of many total variation based image priors, i.e. TV [109], BTV [63], NLTV [111], and BSWTV [112]. In the vector form, Eq. 4.11 can be rewritten as

$$\mathrm{R}(\mathbf{x}) = \sum_{\mathbf{v}\in\mathcal{N}(\mathbf{u})} \|W_{\mathbf{d}} \odot (S_{\mathbf{d}} - \mathrm{I})\mathbf{x}\|_1, \quad \mathbf{d} = \mathbf{u} - \mathbf{v}, \tag{4.12}$$

where $S_{\mathbf{d}} \in \mathbb{R}^{p\times p}$ denotes the shifting matrix which shift $\mathbf{x}$ by $\mathbf{d}$ (in 2D coordinate), $\odot$ denotes the Hadamard product. Weighting functions are assembled in weighting matrix $W_{\mathbf{d}} = \mathrm{diag}(\mathbf{w}_{\mathbf{d}})$, with $\mathbf{w}_{\mathbf{d}} \in \mathbb{R}^p$. The main advantage of this regularization term is the flexibility in defining a weighting function to capture the unique properties of the SR problem. For example, setting $\mathcal{N}(\mathbf{u})$ to direct neighborhood and the weighting to a constant gives us TV [109] which regularizes the local smoothness between adjacent pixels. Setting weighting to a function of the pixel distance gives us BTV [63], which assumes that the smoothness is spatially dependent. Another weighting scheme based on bilateral spectrum used in [112] provides a successful regularization for mixed Gaussian-Poisson noise images. Considering the 4D LF data, we proposed a discontinue-aware weighting scheme which assembles three data properties, i.e., spatial distance, edge and occlusion feature,

$$\mathbf{w}_{\mathbf{d}} := w_d \mathbf{w}_e \odot \mathbf{w}_o, \quad w_{\mathbf{d}} \in \mathbb{R}, \mathbf{w}_e, \mathbf{w}_o \in \mathbb{R}^p, \tag{4.13}$$

where the spatial weight $w_{\mathbf{d}} := \exp\left(\frac{\|\mathbf{d}\|_2^2}{\sigma_s}\right)$ adjusts the impact of weighting w.r.t. the relative distance $\mathbf{d}$ and provide a bilateral filtering effect. The edge weight $\mathbf{w}_e := \exp\left(\frac{\|\nabla\mathbf{x}\|_2^2}{\sigma_e}\right)$ and the occlusion weight $\mathbf{w}_o$ penalize the smoothness at image discontinuing area follows a similar weighting strategy as discussed in Sec. 2.3.4.

4.3 Optimization Approach

Combining the data-fidelity term and regularization term discussed in the previous section, we finalize the minimization problem with the following cost function

$$\mathrm{J}(\mathbf{x}) = \lambda_1 \sum_{k=1}^{s_k} \|A_k\mathbf{x} - \mathbf{y}_k\|_1 + \lambda_2 \sum_{k=1}^{s_k} \|A_k\mathbf{x} - \mathbf{y}_k\|_2^2 + \sum_{d=1}^{s_d} \|W_d \odot (S_d - \mathrm{I})\mathbf{x}\|_1, \tag{4.14}$$

Although non-smooth, the cost function is convex, and the existence of the global minimized solution is guaranteed. There are many algorithms that can be used to optimize it. One of the traditional approaches for solving this problem is applying a first-order iterative algorithm such as the steepest gradient descent. A more recent approach is the alternating direction method of multipliers (ADMM) [113], which breaks a complex optimization problem into smaller sub-problems, each can be solved in a simpler manner. Although

ADMM requires more computation for each iterative step as compared to gradient descent, we notice that the overall computation of ADMM is much less considering the similar minimization threshold. In the following, we discuss the use of these approaches for solving the Eq. 4.14.

4.3.1 First-order Iterative Optimization Algorithm

This approach iteratively steps in the steepest descent direction of the objective function, which is the opposite direction of the gradient at the current point. Gradient descent requires the computation of the first-order derivative of the energy function $\mathrm{J}(\mathbf{x})$. However, ℓ^1 norm is non-smooth and it is not differentiable at 0. Instead of computing the gradient, we consider the sub-gradient of ℓ^1 function [114] which is the sign function defined as follows

$$\mathbf{sgn}(\mathbf{x}) := \nabla \left\|\mathbf{x}\right\|_1, \quad [\mathbf{sgn}(\mathbf{x})]_i := \begin{cases} 1, & \mathbf{x}_i > 0, \\ -1, & \mathbf{x}_i < 0, \\ 0, & \mathbf{x}_i = 0, \end{cases} \tag{4.15}$$

The gradient of the cost function in Ep. 4.14 can be derived as

$$\nabla \mathrm{J}(\mathbf{x}) = \sum_{\rho \in \{1,2\}} \sum_{k=1}^{s_k} \lambda_\rho A_k^\mathsf{T} \mathbf{sgn}(A_k\mathbf{x} - \mathbf{y}_k) \odot |A_k\mathbf{x} - \mathbf{y}_k|^{\rho-1} + \sum_{d=1}^{s_d} W_d^\mathsf{T} \odot (S_d^\mathsf{T} - \mathrm{I}) \mathbf{sgn}\Big(W_d \odot (S_d - \mathrm{I})\mathbf{x}\Big), \tag{4.16}$$

As soon as the gradient of the cost function is computed, the value of $\mathbf{x}^{(n+1)}$ in the next iterative step can be updated by simply scaling the gradient by a constant β, referred to as a step size, and added up to $\mathbf{x}^{(n)}$,

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} + \beta \nabla \mathrm{J}_\rho(\mathbf{x}^{(n)}), \tag{4.17}$$

here the step size β plays a vital role in deciding how fast the optimization will converge to the global minimum solution. Choosing a step size that is too-large can lead to an osculation around the minimum and result in a solution that is even worse than the solution from the current iterative step. However, choosing a too-small step size will lead to more iteration, which eventually demands more computation and solving time. Finding the right step size would involve solving for β, which minimizes the cost function for the next update of $\mathbf{x}$,

$$\arg\min_\beta \mathrm{J}_\rho(\mathbf{x}^{(n)} + \beta \nabla \mathrm{J}_\rho(\mathbf{x}^{(n)})).$$

Algorithm 2: Backtracking line search

```
Input: $\beta_0, \gamma, v$
Output: $\beta$
1  $\delta := v < \nabla \mathrm{J}_\rho(\mathbf{x}), \nabla \mathrm{J}_\rho(\mathbf{x}) >$
2  for $k$ in $0, 1, 2, \ldots$ do
3      $\Delta := \mathrm{J}_\rho(\mathbf{x}) - \mathrm{J}_\rho\big(\mathbf{x} + \beta_k \nabla \mathrm{J}_\rho(\mathbf{x})\big)$
4      if $\Delta < \delta\beta^{(k)}$ then
5          $\beta^{(k+1)} := \gamma\beta^{(k)}$
6      else
7          break
8      end
9  end
10 return $\beta^{(k)}$
```

This approach is referred to as an exact line search which is generally infeasible to solve due to the cost of computation per iteration. An alternative is backtracking line search [115] which starts with a relatively large step size and repeatedly shrinks it by a factor $\gamma \in (0, 1)$ until the Armjo-Goldstein condition is fulfilled,

$$\mathrm{J}_\rho(\mathbf{x}) - \mathrm{J}_\rho\big(\mathbf{x} + \beta^{(k)}\big) \geq \beta^{(k)}\delta,$$

where $\beta^{(k)} = \gamma\beta^{(k-1)}$, and δ represents the local slop of the function of β along the search direction $-\nabla \mathrm{J}_{rho}(\mathbf{x})$. The pseudo-code of backtracking line search is listed in Algorithm 2. The two parameters $\gamma, v \in (0, 1)$ are for controlling the update speed and $\beta^{(0)}$ is the upper-bound of the step size.

4.3.2 Alternating Direction Method of Multiplier - ADMM

The main advantage of ADMM is the ability of transferring a complicated optimization problem into simpler sub-problems. In our case, ADMM allows us to reform the $\ell^p - \ell^1$ problem into ℓ^2 and proximal problems which can be solved effectively. In the following, we are going to discuss the application of ADMM for the two type of data fidelity terms.

We start by rewriting the objective function into a more compact form,

$$\mathrm{J}(\mathbf{x}) = \|A\mathbf{x} - \mathbf{b}\|_2^2 + \|F\mathbf{x} - \mathbf{b}'\|_1, \tag{4.18}$$

the matrices F, A and columns vectors $\mathbf{b}$, $\mathbf{b}'$ are defined as in Eq. 4.19. Notice that α_1 and α_2 are absorbed into the matrices and column vectors for simplifying the notation. The size of A, F, $\mathbf{b}$ and $\mathbf{b}'$ are respectively $qs_k \times p$, $(qs_k + ps_d) \times p$, $qs_k \times 1$ and $(qs_k + ps_d) \times 1$. All transformation matrices (A_k and S_d) and weighting matrices (W_d) are assembled into A and F. Low-resolution images b_k are stacked into $\mathbf{b}$ along with padding zeros. O_{ps_d} is zero vector with the size of $ps_d \times 1$.

$$A := \sqrt{\lambda_2}\begin{bmatrix} A_1 \\ A_2 \\ \cdots \\ A_{s_k} \end{bmatrix}, \mathbf{b} := \sqrt{\lambda_2}\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \cdots \\ \mathbf{y}_{s_k} \end{bmatrix},\ F := \begin{bmatrix} \frac{\lambda_1}{\sqrt{\lambda_2}} A \\ S \end{bmatrix}, \mathbf{b}' := \begin{bmatrix} \frac{\lambda_1}{\sqrt{\lambda_2}} \mathbf{b} \\ O_{ps_d} \end{bmatrix}, S := \begin{bmatrix} W_1 \odot (S_1 - I) \\ W_2 \odot (S_2 - I) \\ \cdots \\ W_{s_d} \odot (S_{s_d} - I) \end{bmatrix} \tag{4.19}$$

Taking the compact representation, we rewrite the optimization problem in Eq. 4.14 into the form of ADMM problem,

$$\begin{aligned} \underset{\mathbf{x},\mathbf{z}}{\text{minimize}} \quad & \|A\mathbf{x} - \mathbf{b}\|_2^2 + \|\mathbf{z}\|_1 \\ \text{subject to} \quad & F\mathbf{x} - \mathbf{z} = \mathbf{b}', \end{aligned} \tag{4.20}$$

with the augmented Lagragian reads,

$$\begin{aligned} \mathcal{L}_\vartheta(\mathbf{x}, \mathbf{z}, \mathbf{w}) := {} & \|A\mathbf{x} - \mathbf{b}\|_2^2 + \|\mathbf{z}\|_1 \\ & + \mathbf{w}^\intercal (F\mathbf{x} - \mathbf{z} - \mathbf{b}') + \frac{\vartheta}{2} \|F\mathbf{x} - \mathbf{z} - \mathbf{b}'\|_2^2 \end{aligned} \tag{4.21}$$

The ADMM problem, Eq. 4.20, is then broken into the following sub-problems for the two unknowns $\mathbf{x}$ and $\mathbf{z}$.

$$\begin{aligned} \mathbf{x}^{(k+1)} &= \arg\min_{\mathbf{x}} \mathcal{L}_\vartheta(\mathbf{x}, \mathbf{z}^{(k)}, \mathbf{w}^{(k)}) && (4.22a) \\ &= \arg\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|_2^2 + \tfrac{\vartheta}{2} \left\| F\mathbf{x} - \mathbf{z}^{(k)} - \mathbf{b}' + \tfrac{\mathbf{w}^{(k)}}{\vartheta} \right\|_2^2 \\ \mathbf{z}^{(k+1)} &= \arg\min_{\mathbf{x}} \mathcal{L}_\vartheta(\mathbf{x}^{(k+1)}, \mathbf{z}, \mathbf{w}^{(k)}) && (4.22b) \\ &= \arg\min_{\mathbf{z}} \|\mathbf{z}\|_1 + \tfrac{\vartheta}{2} \left\| \mathbf{z} - \left(F\mathbf{x}^{(k+1)} - \mathbf{b}' + \tfrac{\mathbf{w}^{(k)}}{\vartheta} \right) \right\|_2^2 \\ \mathbf{w}^{(k+1)} &= \mathbf{w}^{(k)} + \vartheta \left(F\mathbf{x}^{(k+1)} - \mathbf{z}^{(k+1)} - \mathbf{b}' \right) && (4.22c) \end{aligned}$$

The sub-problem of $\mathbf{x}$ (Eq. 4.22a) has the form of a least square approximation problem,

$$\tilde{\mathbf{x}} = \arg\min_{\mathbf{x}} \|G\mathbf{x} - \mathbf{c}\|_2^2, \tag{4.23}$$

with $G = \begin{bmatrix} A \\ \sqrt{\vartheta/2}F \end{bmatrix}$, and $\mathbf{c} = \begin{bmatrix} \mathbf{b} \\ \sqrt{\vartheta/2}\left(\mathbf{z}^{(k)} + \mathbf{b}' - \mathbf{w}^{(k)}/\vartheta\right) \end{bmatrix}$. The sub-problem of $\mathbf{z}$, in Eq. 4.22b, is actually a proximal operator of ℓ^1 function,

$$\mathbf{z}^{(k+1)} = \mathbf{prox}_{\vartheta^{-1}\|\cdot\|_1}\left(F\mathbf{x}^{(k+1)} - \mathbf{b}' + \frac{\mathbf{w}^{(k)}}{\vartheta}\right),$$

which has the following closed form solution

$$\mathbf{z}^{(k+1)} = \left[\left|F\mathbf{x}^{(k+1)} - \mathbf{b}' + \frac{\mathbf{w}^{(k)}}{\vartheta}\right| - \frac{1}{\vartheta}\right]_+ \odot \mathbf{sgn}\left(F\mathbf{x}^{(k+1)} - \mathbf{b}' + \frac{\mathbf{w}^{(k)}}{\vartheta}\right) \tag{4.24}$$

4.3.2.1 Solving Least Square Problem

In previous sections, we have see that the sub-problem of $\mathbf{x}$ has the form of least-square approximation, i.e. $\tilde{\mathbf{x}} := \arg\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|_2^2$, where $A \in \mathbb{R}^{m\times n}, \mathbf{x} \in \mathbb{R}^{n\times 1}$, and $\mathbf{b} \in \mathbb{R}^{m\times 1}$. The objective function can be represented in the form of a convex quadratic function

$$f(\mathbf{x}) = \mathbf{x}^\intercal A^\intercal A\mathbf{x} - 2\mathbf{b}^\intercal A\mathbf{x} + \mathbf{b}^\intercal\mathbf{b},$$

whose solution should satisfy the so-called normal equation (i.e., $\nabla f(x) = 0$)

$$A^\intercal A\mathbf{x} = A^\intercal\mathbf{b}$$

If the columns of A are linearly independent, the least-square approximation has the unique solution $\mathbf{x} = (A^\intercal A)^{-1}A^\intercal\mathbf{b}$. However, it is normally the case that its columns are linearly dependent and the positive semi-define matrix $A^\intercal A$ is singular. In such a case, the inverse of $A^\intercal A$ does not exist. In addition, a direct computation of $(A^\intercal A)^{-1}$ is not feasible due to the size and sparsity of A. For these reasons, iteratively solving the least-square approximation problem is a preferable choice. In this work, we employ a well-known conjugate gradient approach on normal equation [115] which guarantees the convergence to the minimum least-square even in the case of singular matrix [116]. Algorithm 3 presents the pseudo-code of the conjugate gradient descent algorithm on the normal equation.

Algorithm 3: Conjugate Gradient

```
Input: $A, \mathbf{b}, \mathbf{x}_0$
Output: $\mathbf{x}$
$r_0 := A^\top \mathbf{b} - A^\top A\mathbf{x}_0$
$p_0 := r_0$
$\pi_0 := r_0^\top r_0$
for $k$ in $0, 1, ..N$ do
    if $\pi_k < T$ then
        break
    end
    $\alpha_k := p_k^\top A^\top A p_k$
    $\mathbf{x}_{k+1} := \mathbf{x}_k + \frac{\pi_k}{\alpha_k} p_k$
    $r_{k+1} := r_k - \frac{\pi_k}{\alpha_k} A^\top A p_k$
    $\pi_{k+1} := r_{k+1}^\top r_{k+1}$
    $p_{k+1} := z_{k+1} + \frac{\pi_{k+1}}{\pi_k} p_k$
end
```

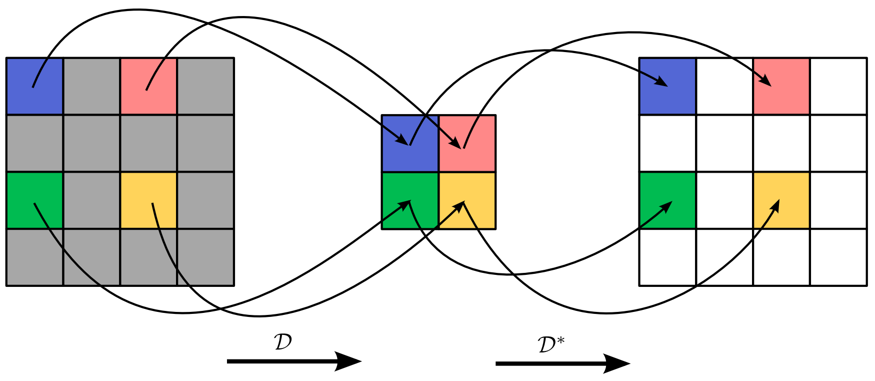


Figure 4.4: Implementation of Downsampling Operator

4.3.2.2 Treatment of Linear Operators

All computations are eventually broken down to matrix multiplication for which the largest computational efforts are on $A_k, S_d, \quad k \in [1, s_k], d \in [1, s_d]$, and their adjoint versions $A_k^\top, S_d^\top$. These matrixes are practically very large and sparse. For example, given a pair of low-resolution and high-resolution: $s_x \times s_y := 128 \times 128$ and $s_X \times s_Y = 512 \times 512$ (i.e., $4\times$ super-resolution). Assuming $s_k = 16$ and $s_d = 8$, the size of A is $2^{18} \times 2^{18}$ and the size of S is $2^{21} \times 2^{18}$. Direct computation of these matrices is infeasible. Therefore, we decide to implement these matrices in the form of linear functions of 2D variables instead of sparse matrices and vectorized inputs. Table 4.1 shows the mapping of matrix to linear function on 2D array.

Table 4.1: Implementation of sparse matrix multiplication as linear operators

Matrix	Function
$A_k\mathbf{x}$	$\mathcal{D} \circ \mathcal{B} \circ \mathcal{W}_k(X)$
$A_k^{\mathsf{T}}\mathbf{x}$	$\mathcal{W}_k^* \circ \mathcal{B}^* \circ \mathcal{D}^*(X)$
$S_k\mathbf{x}$	$\mathcal{S}_{\mathbf{u}}(X)$
$S_k^{\mathsf{T}}\mathbf{x}$	$\mathcal{S}_{-\mathbf{u}}(X)$

For downsampling operator $\mathcal{D}$, a simple resampling scheme is employed as depicted in Fig. 4.4. For each block of $\zeta_x \times \zeta_y$ pixels, one pixel at the top-left location is picked and put into the low-resolution grid. The adjoint operator $\mathcal{D}^*$ is therefore simply putting back the corresponding pixel to this location. The bluring operator $\mathcal{B}$ is modelled by a simple Gaussian kernel with a standard deviation of $\sigma = \frac{1}{4}\sqrt{\zeta^2 - 1}$ and a size of 3σ as suggested in [117]. The warping operator $\mathcal{W}_k$ and its adjoint operator $\mathcal{W}_k^*$ are implemented as forward-warping function and approximated as reverse warping function respectively. These functions are associated with a set of disparity maps at each of the perspectives employed for super-resolution. Assumes that a set of s_k low-resolution sub-aperture images each with its perspective index is in $P = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_{s_k}\}$ are inputs to estimate an super-resoltion image at $\boldsymbol{\theta}_0 \in P$. For each perspective $\boldsymbol{\theta}_k$, we need to find the disparity map $\boldsymbol{\omega}_k$. The forward warping function $\mathcal{W}_k$ will warp the SAI from perspective $\boldsymbol{\theta}_0$ to $\boldsymbol{\theta}_k$ using $\boldsymbol{\omega}_k$, i.e., $\widehat{\boldsymbol{L}}(\mathbf{z}, \boldsymbol{\theta}_k) = \boldsymbol{L}(\mathbf{z} + \boldsymbol{\theta}_k\boldsymbol{\omega}_k, \boldsymbol{\theta}_k)$, while the reverse warping function $\mathcal{W}_k^*$ will warp the input SAI from perspective $\boldsymbol{\theta}_k$ to $\boldsymbol{\theta}_0$ using $\boldsymbol{\omega}_0$, i.e., $\widehat{\boldsymbol{L}}(\mathbf{z}, \boldsymbol{\theta}_0) = \boldsymbol{L}(\mathbf{z} + \boldsymbol{\theta}_0\boldsymbol{\omega}_0, \boldsymbol{\theta}_0)$.

The transformation matrix S can be implemented in the form of weighted directional gradient ($\nabla^{U,V}$) computed for a direction set $U = \{\mathbf{d}_i | \mathbf{d}_i \in \mathbb{N}^2, i = 1, .., s_d\}$ and a weight set $V = \{V_i | V_i \in \mathbb{R}^{s_X \times s_Y}, i = 1, .., s_d\}$. Let I be the SAI at perspective $\boldsymbol{\theta}_0$, $I(\mathbf{z}) = \boldsymbol{L}(\mathbf{z}, \boldsymbol{\theta}_0)$, we computed $\nabla^{U,V} I$ as follow,

$$\boldsymbol{G} = \nabla^{U,V} I = \left(\frac{\partial}{\partial \mathbf{d}_1}, \frac{\partial}{\partial \mathbf{d}_2}, .., \frac{\partial}{\partial \mathbf{d}_{s_d}} \right) I,$$

with the weighted directional derivative $\partial/\partial\mathbf{d}_i$ approximated by finite differences,

$$G_{d_i}(\mathbf{z}) = \frac{\partial}{\partial \mathbf{d}_i} I(\mathbf{z}) = V_i(\mathbf{z}) \left(I(\mathbf{z}) - I(\mathbf{z} + \mathbf{d}_i) \right).$$

The adjoint matrix S^{T} is then computed in the form of weighted directional divergence,

$$\mathtt{div}^{U,V} \boldsymbol{G} = \nabla^{U,V} \cdot \boldsymbol{G} = \sum_{i=1}^{s_d} \frac{\partial G_{\mathbf{d}_i}}{\partial \mathbf{d}_i}.$$

4.4 GPU-Accelerated Strategy

This section discusses the proposed GPU-accelerated architectures for 4D light field super-resolution. The acceleration is achieved by the mean of parallel computation using graphics processing units. OpenCL library is employed in this work to implement proposed architectures. OpenCL was selected over CUDA because of its cross-platform compatibility. The source codes using OpenCL can be compiled and run on different platforms, i.e. AMD GPU, NVIDIA GPU. For the super-resolution of 4D light field images, the selected algorithm is first decomposed into a set of processing functions. These functions are then analyzed to justify their capability of parallel computation. This information is then used to decide the global work size of each GPU kernel. In the end, all the functions are transformed into OpenCL kernel code and executed for each work-item.

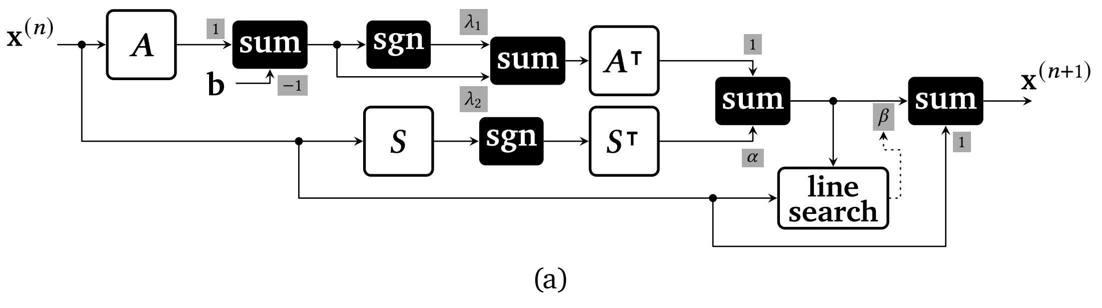
4.4.1 Accelerated Steepest Gradient Descent

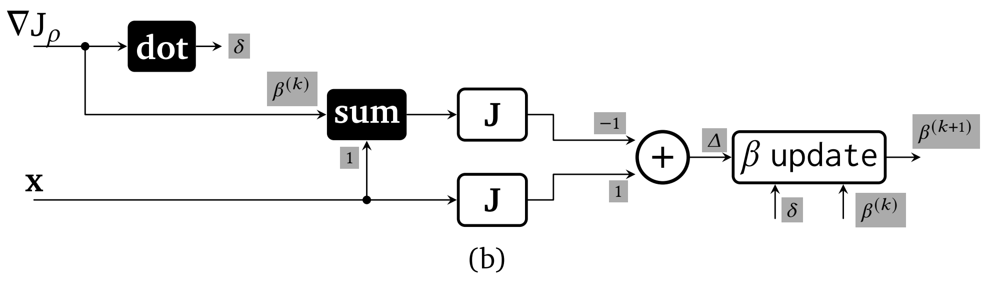
For the steepest gradient descent approach the main computation task is the calculation of the gradient in Eq. 4.16, which can be rewritten in the following compact form,

$$\nabla \mathrm{J}(\mathbf{x}) = A^{\intercal}\left(\lambda_1 \mathbf{sgn}(A\mathbf{x} - \mathbf{b}) + \lambda_2(A\mathbf{x} - \mathbf{b})\right) + S^{\intercal}\mathbf{sgn}(S\mathbf{x}), \tag{4.25}$$

with A, $\mathbf{b}$, and S are formed by respectively stacking the transformation matrices A_k, the low-resolution inputs $\mathbf{y}_k$, and the weighted differential matrices $W_d \odot (S_d - I)$, $k = 1, ..., s_k$, $d = 1, ..., s_d$. The overall computation flow of a gradient descent iterative step is depicted in Fig. 4.5(a). In the figure, OpenCL kernels are denoted by a black background box. Starting with the computed HR solution $\mathbf{x}^{(n)}$ from the previous iterative step, we follow Eq. 4.25 to compute the gradient $\nabla \mathrm{J}(\mathbf{x}^{(n)})$. An optional backtracking line-search is applied to estimate a step size β before updating the HR solution $\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} + \beta \nabla \mathrm{J}(\mathbf{x}^{(n)})$.

Figure 4.5(b) shows the computation flow of backtracking line search following Algorithm. 2. Before starting the main loop searching for a suitable step size, we compute the scaling factor δ by applying a dot product kernel. The objective function of the current HR solution $(\mathrm{J}(\mathbf{x}^{(n)}))$ needs to be computed only one time for each line search task. Each iteration includes a sum kernel to compute an intermediate HR solution advancing in the search direction and an objective function. The β`update` function decides between updating β and stop the loop, Algorithm 2 line 4-8. The objective function Eq. 4.14 requires the application of matrices A, S, a sum kernel and three norm kernels as shown in Fig. 4.5(c). It is also noticed that the norm kernels return a scalar. Since it is not an advantage to compute scalar operations on GPU, we execute these operations on CPU. This includes the





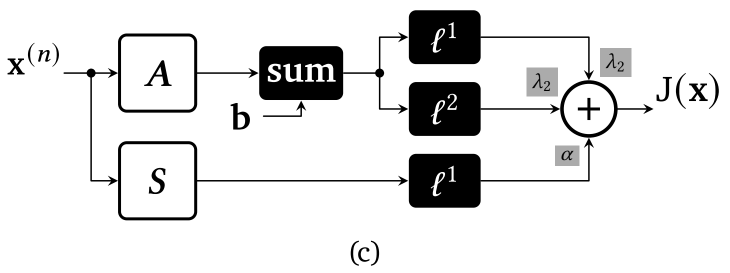


Figure 4.5: Block digram representation of proposed GPU-based light field image super-resolution. (a) Overall computation of one steepest descent step. (b) line-search. (c) Computation of the objective function J

scalar sum, denoted by $\oplus$, and β `update` function, Fig. 4.5(b). The implementation of transformation matrices A, S and their adjoint versions $A^\top$, $S^\top$ are going to be discussed in the next section.

4.4.2 ADMM Solver

This section discusses the OpenCL realization of optimization strategy presented in Sec.4.3.2. For a better handling of the computation flow, we did the following modifications to ADMM iteration in Eq. 4.22. First, the order of sub-problems is rearranged such that **x**-step comes after **z**-step and **w**-step. This way allows us to make use of the computation of $F\mathbf{x}$ for all sub-problems. Secondly, the parameter ϑ is absorbed into **w** (i.e., **w** instead of $\mathbf{w}/\vartheta$) to save unnecessary scalar multiplications. ϑ only takes part in the computation of the proximal operator (z-step) and solving of the least square problem (x-step). Fig. 4.6

illustrates the modified computations of ADMM solver which is also listed in Algorithm 4.

Algorithm 4: Minimization of the cost function in Eq. 4.14 with ADMM iterative solver.

```
Input: $\vartheta, \mathbf{x}_0, N$
Output: $\mathbf{x}$
$\mathbf{x}^{(0)} := \mathbf{x}_0$
$\mathbf{w}^{(0)} := 0$
for $n$ in $1, 2, ..N$ do
    $\mathbf{a} := A\mathbf{x}^{(n-1)} - \mathbf{b}$
    $\mathbf{u} := F\mathbf{x}^{(n-1)} - \mathbf{b}' + \mathbf{w}^{(n-1)}$
    $\mathbf{z}^{(n)} := \mathbf{prox}_{\rho^{-1}\|\cdot\|_1}(\mathbf{u})$          ▷ Solving Eq. 4.22b
    $\mathbf{w}^{(n)} := \mathbf{u} - \mathbf{z}^{(n)}$          ▷ Computing Eq. 4.22c
    $\mathbf{f} := 2\mathbf{w}^{(n)} - \mathbf{w}^{(n-1)}$
    $\mathbf{v} := A^\intercal \mathbf{a} + \frac{\vartheta}{2} F^\intercal \mathbf{f}$
    $\mathbf{x}^{(n)} :=$ xstep$(\mathbf{v}, \mathbf{x}^{(n-1)})$          ▷ Solving Eq. 4.22a
end
return $\mathbf{x}^{(n)}$
```

The ADMM solver takes in three arguments, the parameter ϑ, an initial guess ($\mathbf{x}_0$) and the number of iterations (N), as in Algorithm 4. Before the iteration, we initialized $\mathbf{x}$ with $\mathbf{x}_0$, a bi-cubic up-sampling of the low-resolution image, and $\mathbf{w}$ with zeros, line 1, 2. Each iteration starts with the computation of $A\mathbf{x}$ and $F\mathbf{x}$ which are associated to ℓ^2 and ℓ^1 terms of the objective function, Eq. 4.18. While $A\mathbf{x}$ is subtracted by $\mathbf{b}$, line 4, $F\mathbf{x}$ is subtracted by $\mathbf{b}'$ and summed with $\mathbf{w}$, line 5. Since F is a stack of A and S and $\mathbf{b}'$ includes $\mathbf{b}$, Eq. 4.19, we avoid the re-computation of $A\mathbf{x} - \mathbf{b}$ by extracting it from $F\mathbf{x} - \mathbf{b}'$ as depicted in Fig. 4.6. The sum and subtract operations in line 5 are realized by two-arguments sum kernels (i.e., `sum` in Fig. 4.6). The gray box attached to each input to the sum kernel denotes the scalar scaling of the input. On line 6, we conduct a $\mathbf{z}$-step by computing the proximal operator of $\mathbf{u}$. This proximal operator is realized by an OpenCL kernel `prox`, as in Fig. 4.6, followed by a sum kernel which realizes $\mathbf{w}$-step, line 7 Algorithm 4.

After the computation of $\mathbf{z}$ and $\mathbf{w}$, the next step is preparing the residual input for the conjugate gradient descent solver in $\mathbf{x}$-step, $\mathbf{v} = G^\intercal(G\mathbf{x} - \mathbf{c})$. From Eq. 4.23, we

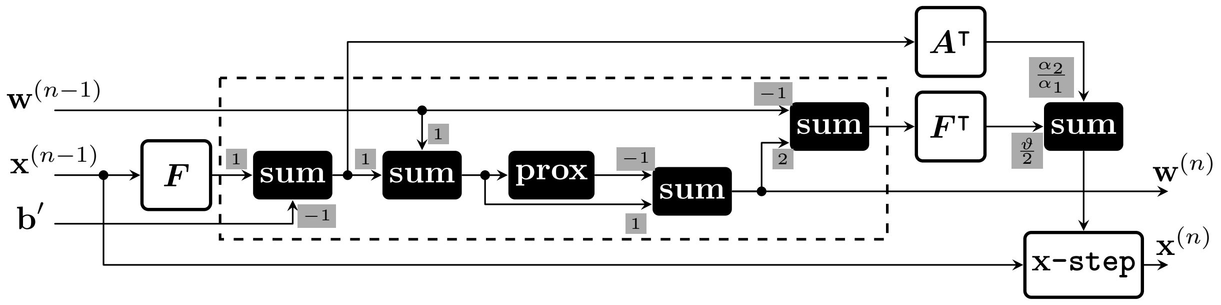


Figure 4.6: Computation flow of one ADMM iteration.

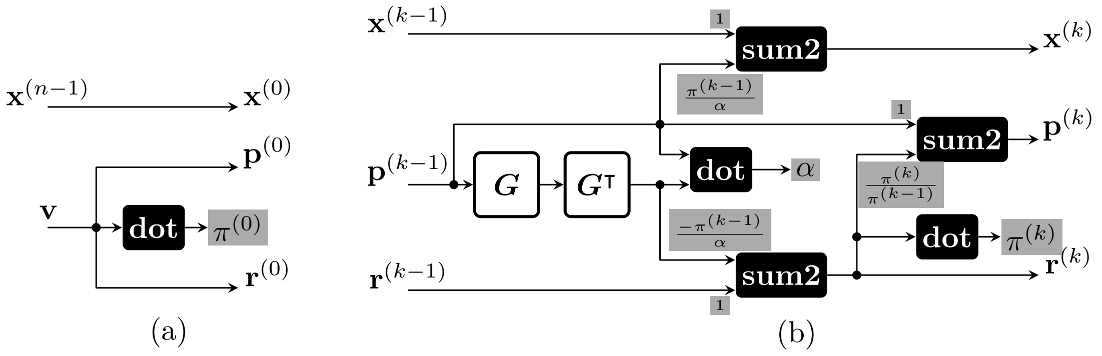


Figure 4.7: Computation flow of xstep. (a) Pre-computation; (b) Conjugate gradient descent iteration.

have

$$
\begin{aligned}
\mathbf{v} = & \begin{bmatrix} A \\ \sqrt{\vartheta/2}F \end{bmatrix}^T \begin{bmatrix} A\mathbf{x} - \mathbf{b} \\ \sqrt{\vartheta/2}\left(F\mathbf{x} - \mathbf{z}^{(n)} - \mathbf{b}' + \mathbf{w}^{(n)}\right) \end{bmatrix} \\
= & A^\intercal(A\mathbf{x} - \mathbf{b}) + \vartheta/2F^\intercal(F\mathbf{x} - \mathbf{z}^{(n)} - \mathbf{b}' + \mathbf{w}^{(n)}) \\
= & A^\intercal\mathbf{a} + \frac{\vartheta}{2}F^T\mathbf{f}
\end{aligned}
\tag{4.26}
$$

With the computation of **f**, Algorithm 4 line 8, as $\mathbf{f} = 2\mathbf{w}^{(n)} - \mathbf{w}^{(n-1)} = \mathbf{u} - \mathbf{z} - \mathbf{w}^{(n-1)} + \mathbf{w}^{(n)} = F\mathbf{x}^{(n-1)} - \mathbf{z} - \mathbf{b}' + \mathbf{w}^{(n)}$. The computations of **f** and **v** are realized by two sum kernels directly before and after F^T as in Fig. 4.6. Notice that we made a scaling of $A^T\mathbf{a}$ by $\frac{\alpha_2}{\alpha_1}$ since the **a** is extracted from $F\mathbf{x} - \mathbf{b}'$ which has a different scalar scaling of matrix A and column vector **b**. Another note from the implementation of Fig. 4.6 is that the group of OpenCL kernels marked by a dashed rectangle would be combined into a single kernel, since these kernels share element-wise operators.

As discussed in the previous section, conjugate gradient descent on normal equation is employed to solve ℓ^2 optimization problem of **x**-step. Fig. 4.7 depicts the computation flow of **x**-step, while its pseudo code is listed in Algorithm 5. There are two inputs, i.e. **v**, $\mathbf{x}^{(n-1)}$, and two scalar parameters, i.e., τ, K. The computed HR image from the previous ADMM iteration $\mathbf{x}^{(n-1)}$ is used as the initial guess for the conjugate gradient descent solver, while the residual **v** is used to initialize $\mathbf{r}^{(0)}$, $\mathbf{p}^0$ and compute the initial error $\pi^{(0)}$. The two parameters τ and K specify the error threshold and the maximum number of conjugate gradient iterations, respectively. The stop condition is that either the residual **r** is sufficiently small or the maximum number of iterations is reached, Algorithm 5 line 4,5. All computations in Algorithm 5 can be effectively broken down into GPU kernel implementation. Beside the forward and backward transform $(G, G^\intercal)$, there are two kernels `sum` and `dot`, as in Fig. 4.7, which represents element-wise sum and dot product respectively.

Algorithm 5: Solving **x**-step

```
Input: $\mathbf{v}, \mathbf{x}^{(n-1)}, \tau, K$
Output: $\mathbf{x}$
1  $\mathbf{x}^{(0)} := \mathbf{x}^{(n-1)}$
2  $\mathbf{p}^{(0)} := \mathbf{r}^{(0)} := \mathbf{v}$
3  $\pi^{(0)} :=< \mathbf{r}^{(0)}, \mathbf{r}^{(0)} >$                ▷ dot product
4  for $k$ in $1, 2, ..K$ do
5      if $\pi^{(k-1)} < \tau$ then
6          $\mathbf{v} := G^\intercal G \mathbf{p}^{(k-1)}$
7          $\alpha :=< \mathbf{v}, \mathbf{p}^{(k-1)} >$
8          $\mathbf{r}^{(k)} := \mathbf{r}^{(k-1)} - \frac{\pi^{(k-1)}}{\alpha} \mathbf{v}^{(k-1)}$
9          $\pi^{(k)} :=< \mathbf{r}^{(k)}, \mathbf{r}^{(k)} >$
10         $\mathbf{p}^{(k)} := \mathbf{p}^{(k-1)} + \frac{\pi^{(k)}}{\pi^{(k-1)}} \mathbf{r}^{(k-1)}$
11         $\mathbf{x}^{(k)} := \mathbf{x}^{(k-1)} + \frac{\pi^{(k-1)}}{\alpha} \mathbf{p}^{(k-1)}$
12     end
13 end
14 return $\mathbf{x}^{(k)}$
```

From the Eq. 4.19 and Eq. 4.23, we can derive the computation of $G^T G$ in the form of A and S as

$$G^T G = A^T A + \frac{\vartheta}{2} F^\intercal F = \left(1 + \frac{\vartheta}{2} \frac{\alpha_1^2}{\alpha_2}\right) A^T A + \frac{\vartheta}{2} S^T S,$$

with the kernel realization of A,S and its adjoint version $A^\intercal$, $S^\intercal$ shown in Fig. 4.8. The figure illustrates the change in the size of the column vector after each kernel execution. Regarding Fig. 4.8, `fwarp`, `bwarp`, `blur`, `up`, and `down` denote the forward warp, reverse warp, blur, up-sampling and down-sampling kernel respectively. `wdg` kernel realizes the

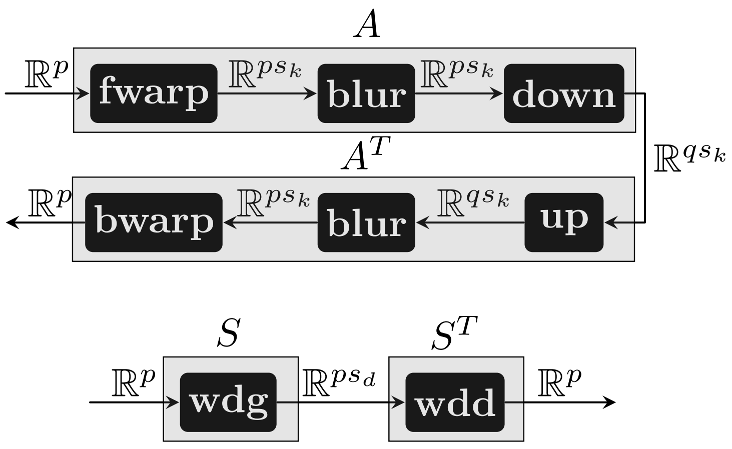


Figure 4.8: Kernel realization of $A, A^\intercal, S, S^\intercal$

weighted directional gradient (i.e., $\nabla^{U,V}$), while the weighted directional divergence (i.e., $\mathtt{div}^{U,V}$) is implemented by wdd kernel.

For each kernel execution, a global (work-group) size and a local (work-group) size need to be decided. The global size specifies the number of work-groups that need to be scheduled, and the local size specifies the number of work-items associated with one work-group. They are typically defined in the form of a 2-tuple parameter consisting of horizontal and vertical size. The global size needs to be a multiple of the local size. In this work, we apply a common practice that divides the computation workload by the number of pixel locations in the output. In other words, each work-item will be in charge of calculating the output value for one particular pixel location. Let $\kappa = (\kappa_x, \kappa_y)$ is a local work-group size, where κ_x and κ_y denote the number of work-items in x and y direction respectively. Each work-group contains $\kappa_x\kappa_y$ work-items and computes output value for a patch of size $\kappa_x \times \kappa_y$. In our work, most OpenCL kernels have an array of 2D images as output. For example, the forward warp kernel (`fwarp`) generates s_k HR image data and weighted direction gradient kernel (`wdg`) results in s_d HR image data. To make use of the Image2D object, we tile these images in the form of a 2D array. Suppose n_x, n_y, and s are respectively the width, the height of the image, and number of images, global size will be set to $\left(\{n_x\}_{\kappa_x,s}, \{n_y\}_{\kappa_y,s} \right)$, where operator $\{\bullet\}_{\kappa,s}$ is defined as

$$\{n\}_{\kappa,s} = \kappa \left\lceil \frac{n\lceil \sqrt{s} \rceil}{\kappa} \right\rceil, \quad \kappa, n, s \in \mathbb{N}^+. \tag{4.27}$$

Table 4.2 reports the global work-group size of implemented kernels in Fig. 4.8. As discussed in Sec.4.3.2.2, the transform matrix is realized in the form of linear operators. The column vector is actually realized as a 2D array stored with OpenCL Image2D object. The forward

Table 4.2: Global work size of GPU kernels

#	Kernel	Horizontal size	Vertical size
1	`up`	$\{s_X\}_{\kappa,s_k}$	$\{s_Y\}_{\kappa,s_k}$
2	`blur`		
3	`fwarp`		
4	`bwarp`	$\{s_X\}_{\kappa,1}$	$\{s_Y\}_{\kappa,1}$
5	`wdd`		
6	`down`	$\{s_x\}_{\kappa,s_d}$	$\{s_y\}_{\kappa,s_d}$
7	`wdg`	$\{s_X\}_{\kappa,s_d}$	$\{s_Y\}_{\kappa,s_d}$

warp operators $\mathcal{W}_k, k \in [1, s_k]$ are implemented with a single OpenCL kernel (i.e., `fwarp`) which warps the 2D input high-resolution image ($s_X \times s_Y$) to s_k perspectives resulting in a tensor ($s_k \times s_X \times s_Y$). The reverse warp operators $\mathcal{W}_k^*, k \in [1, s_k]$ are implemented by kernel `bwarp` which warps s_k images to the perspective of HR image and then apply an element-wise sum of them resulting in a 2D array ($s_X \times s_Y$). In a similar manner, the weighted directional gradient kernel (`wdg`) generates a tensor ($s_d \times s_X \times s_Y$) from input HR image, while kernel `wdd` realizes the weighted directional divergence outputting a 2D array with the same size as HR image. $\mathcal{B}$ and $\mathcal{B}^*$ are implemented with kernel `blur`, while $\mathcal{D}$ and $\mathcal{D}^*$ are realized with kernel `down` and `up` respectively.

4.5 Experimental Results

This section presents the performance evaluation of the proposed approach, which is divided into two parts. In the first part, we analyze the reconstruction quality of the proposed algorithm and the performance of the GPU implementation of the iterative solvers including gradient descent and ADMM. In the second part, we compare our approach with the related works concerning super-resolution and acceleration performances.

4.5.1 Evaluation of LFSR Computational Framework

Light field scenes from 4D synthetic dataset [91] are employed to evaluate the robustness of the SR model and analyze the convergence of iterative solvers. This dataset is selected since it includes plenty of scenery and provides accurate disparity maps. We follow the degradation model discussed in Sec. 4.1 to prepare the input data with two test scaling factors, i.e., $\times 2, \times 4$. The observation noises are parameterized by σ and ν, which respectively denotes the standard deviation of Gaussian noise (i.e., $\mathcal{N}(\mu = 0, \sigma)$) and the percentage of

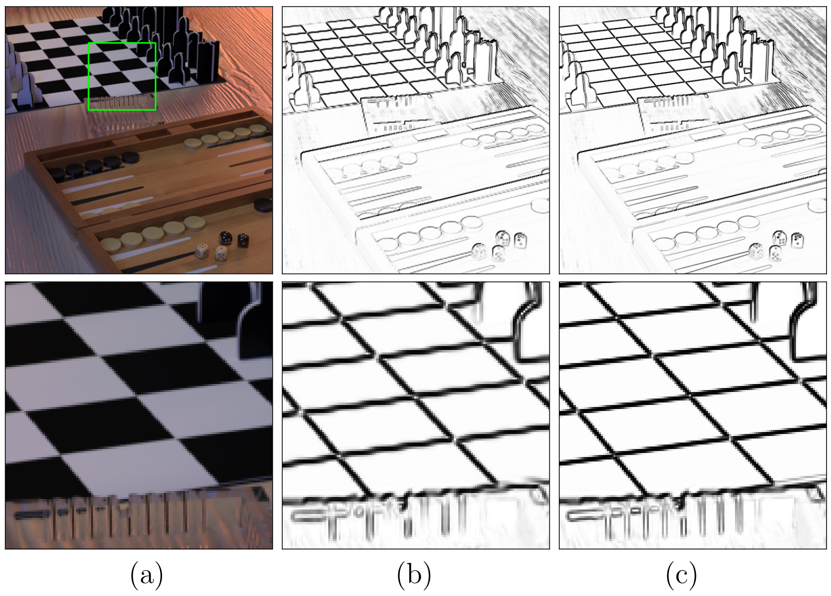


Figure 4.9: Regularization weights calculation for LF scene '*boardgames*'; top row: full size image and weights are; bottom row: zoom-in region marked by blue rectangle; (a) Ground-truth image; (b) weights at the 1st iteration; (c) weights at the 10th iteration

impulse noise (i.e., salt and pepper). In order to match the practical use cases in which the high-resolution disparity maps are not available, the provided disparity maps are down-scaled by the same factor as of the test case (i.e., ×2, ×4) and then are interpolated back to the original size and used in the warping functions.

The regularization weights computed for the scene '*boardgames*' are shown Figure 4.9. It is expected that a strong weighting is applied to the region where high-frequency information is occupied (i.e., texture edges, occlusions). The regularization weight is a combination of spatial weight (w_d), edge weight ($\mathbf{w}_e$), and occlusion weight ($\mathbf{w}_o$). To strengthen the regularizing effect, the weights are recomputed for each ADMM iteration using the current computed super-resolution image $\mathbf{x}$. When the optimization starts, $\mathbf{x}$ is initialized to a bi-cubic up-sampling of the low-resolution image. This explains the blur edges of regularization weights at iteration 1, as shown in Fig. 4.9 (b). However, it could be observed after each ADMM iteration that the qualities of $\mathbf{x}$ and regularization weight are gradually improved. As shown in Fig. 4.9 (c), the regularization weight after 10 ADMM iterations capture well the high-resolution structure of the reconstructed scene.

Fig. 4.10 visualizes the SR result for the ×4 test case of LF scene '*dishes*'. We employed 17 LR sub-aperture images as inputs to calculate the cost function in Eq. 4.14 which is then solved by ADMM iterative solver. The SAIs are picked up from 5×5 angular views in a star-

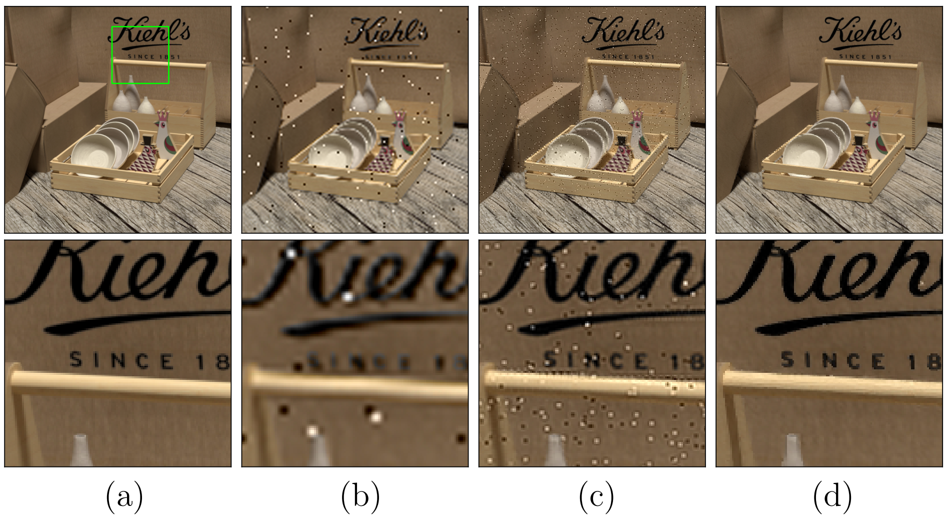


Figure 4.10: $\times 4$ super-resolution of LF scene '*dishes*' ($\sigma = 1$, $\nu = 1\%$); top row: full size image; bottom row: zoom-in of region marked by blue rectangle; (a) Ground-truth image; (b) bi-cubic up-sampling (21.72 dB); (c) 1 ADMM iteration (24.96 dB) (d) 5 ADMM iteration (29.53 dB)

like structure. As compared to the bi-cubic up-sampling image used as an initial solution (Fig. 4.10 (b)), the reconstructed HR image after the first ADMM iteration (Fig. 4.10 (c)) demonstrates an obvious improvement in visibility. Although the noise effect from the combination of multiple SAIs is still visible, it is possible to observe the texture content (i.e., small characters in the middle of the zoom-in region). After 5 ADMM iterations, the noise effect is removed, resulting in a significant enhancement in visual quality with 4.6dB and 7.8dB improvement as compared to the 1st iteration's solution and the initial solution, respectively.

To evaluate the contribution of ℓ^1 and ℓ^2 data terms in reconstructing HR perspective image under mix-noise condition, we prepare a test case in which LR Light field is severely damaged by noise effects, see Fig. 4.11. While keeping the regularization part unchanged, we tuned data fidelity parameters (λ_1, λ_2) to find a solution with the highest PSNR score for each model (i.e., ℓ^1, ℓ^2, $\ell^1 + \ell^2$). We observed that using only ℓ^2 data fidelity tends to oversmooth the solution due to the effect of the ℓ^2 norm. Although ℓ^1 data fidelity well preserves the sharp edge structure, it also carries the effect of the noisy pixels into the solution. The proposed mix-noise data term combines the impacts of both ℓ^2 norm and ℓ^1 norm and provides a better reconstruction quality.

The number of input LR images plays an important role in the quality of the reconstructed HR image. Although demanding higher computation resources, we observed that more input SAIs tend to provide higher reconstruction qualities. Fig. 4.12 reports the $\times 4$ super-

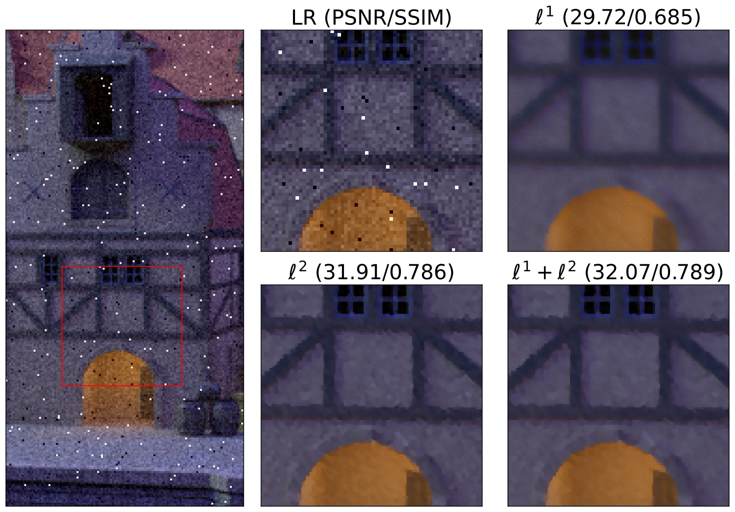


Figure 4.11: ×2 super-resolution of LF scene ‘*medieval2*’ ($\sigma = 10, \nu = 1\%$); *left* : a cropped noisy LR input; *right*: four zoom-in of the marked region from an LR input and three different configurations of data fidelity term.

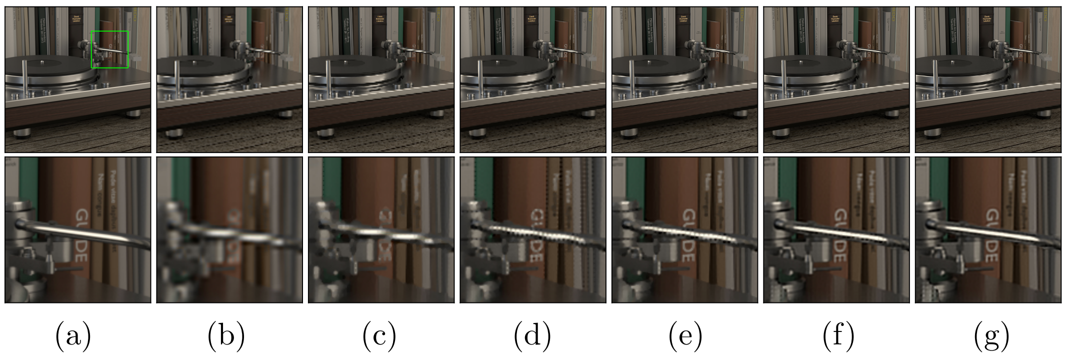


Figure 4.12: Super-resolution ×4 results of LF scene ‘*vinyl*’ under different number of inputs. *top* full size HR image; *bottom* zoom-in of marked region; (a) Ground-truth image; (b) bi-cubic upsampling (24.04dB); (c) 3 SAIs (27.55dB); (d) 5 SAIs (29.33dB); (e) 9 SAIs (30.54dB); (f) 25 SAIs (31.65dB); (g) 49 SAIs (32.09dB).

resolution results of LF scene ‘*vinyl*’ where different numbers of LR sub-aperture images are used. As can be seen from the figure, giving more input images to the computational problem (Eq. 4.14) results in a better visual quality of HR solutions, which is also evident from the reported PSNR scores. Specifically, an improvement of 3.5dB as compared to bi-cubic up-sampling can be achieved with three input images. When increasing the number of LR images to 5, 9, 25, and 49, we observed an incremental gains of 1.8dB, 1.2 dB, 1.1 dB and 0.44 dB, respectively.

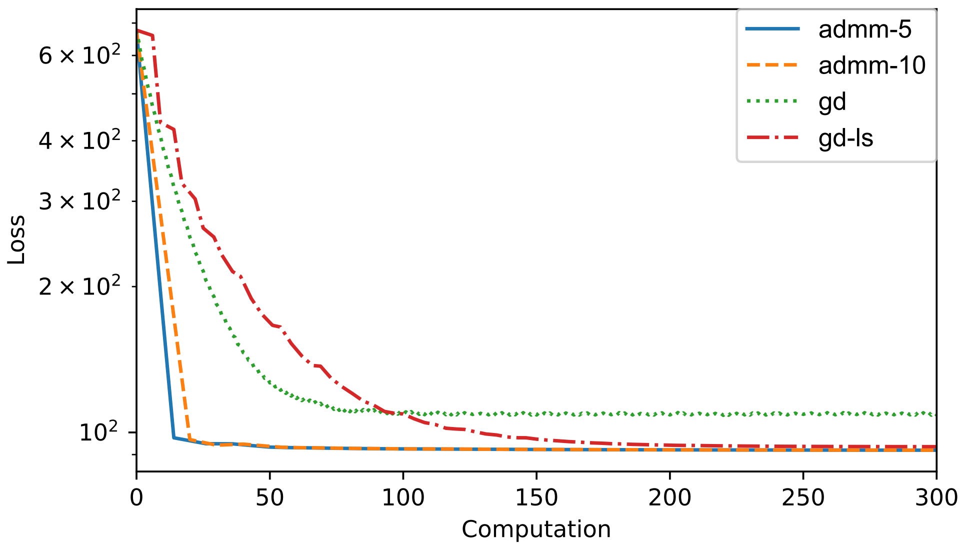


Figure 4.13: Optimization results of different solvers

In order to compare the convergence of the iterative solvers, we employ the matrix transform functions (i.e.,A,S) and their adjoint versions (i.e., $A^\intercal$, $S^\intercal$) as computation units (CU). As derived in Sec. 4.3, these transforms are the most dominant computation tasks and exist in every iterative step. Each CU is either a combination of A and S as for computing the cost function J or $A^\intercal$ and $S^\intercal$ as for computing the gradient ∇J. In this experiment, we built a cost function for ×2 SR problem of LF scene '*vinyl*' and applied four different configurations of the iterative solvers to optimize it. The first two are gradient descent solver (GD) without and with line search denoted as *gd* and *gd-ls* respectively. The last two are ADMM solvers in which we configure the maximum number of conjugate gradient steps to 5 (*admm-5*) and 10 (*admm-10*). Fig. 4.13 presents the plot of the loss function against the accumulated CU. Providing a good step size, GD without line search can make a rapid reduction in the cost function for the first few iterations. However, due to the fixed step size, the GD cannot optimize the loss function further after 80 CUs. In contrast, *gd-ls* seems slow at the beginning due to the search for an appropriate step size but is able to surpass *gd* at around 100 CUs and approach the global minimum after around 300 CUs. Avoiding the costly line-search tasks, both configurations of the ADMM solver demonstrate a superior convergence rate as compared to GD. We also observed that setting the maximum number of conjugate descent steps to 5 does shorten the computation effort for the first few iterations. However, at later iterations when the early stop condition is satisfied, i.e., Algorithm 5 line *5*, both settings result in a similar performance.

The proposed computation framework can also be applied to a more challenging image condition such as motion blur. In such a case, the motion blur can be modelled by a

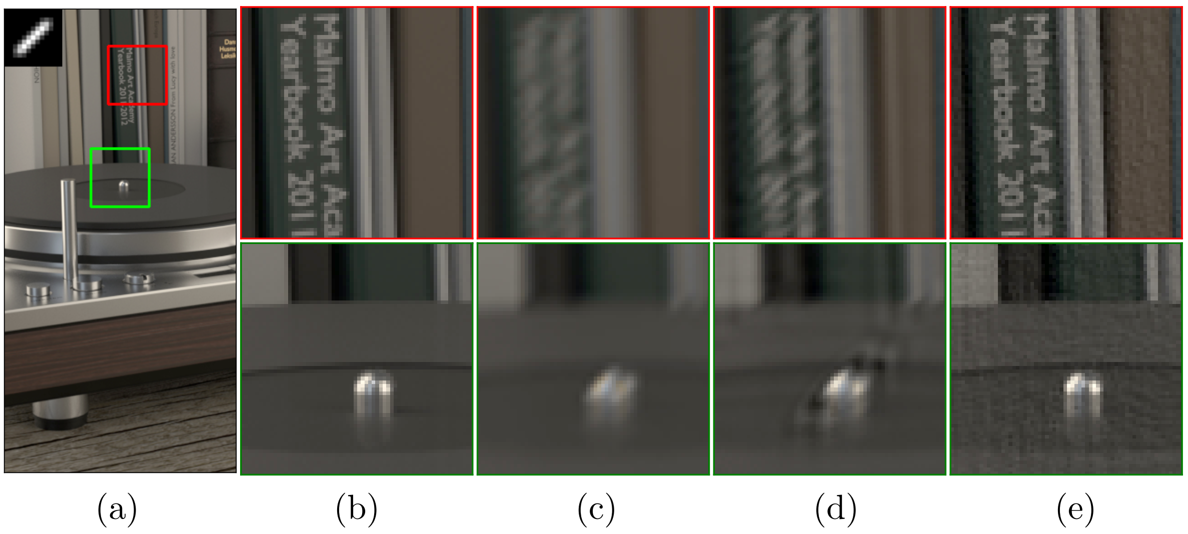


Figure 4.14: ×2 super-resolution result of LF scene 'vinyl' degraded by motion blur. (a) a cropped of ground truth with two marked region and motion blur kernel shown at top left corner; (b) zoom-in of ground truth image; (c) bi-cubic initial image (26.86dB); (d) after 1^{st} ADMM iteration (30.27dB); (e) after 10^{th} ADMM iteration (35.43dB).

convolutional kernel as a realization of the linear operator $\mathcal{B}$ (see Fig. 4.2). Fig. 4.14 shows our ×2 SR result for low-resolution LF input degraded by a 45 degree motion blur. The blur kernel is shown on the top left corner of Fig. 4.14(a) and two zoom-in regions of the bi-cubic upsampling of degraded low-resolution SAI are shown in Fig. 4.14(c). Taking 25 SAIs as inputs to our reconstruction algorithm, we can achieve more than a 3 dB improvement in PSNR score after one ADMM iteration. The high-resolution LF image is well reconstructed after 10^{th} ADMM iterations with clear texture information and motion trace.

4.5.2 Evaluation of GPU-based Gradient Descent Solver

In this section, we discuss the evaluation of the GPU-accelerated gradient descent solver. The scene '*dino*' is employed in this experiment. This synthetic scene contains various sharp-edged objects placed at different depths, hence it is suitable for testing the estimated disparity maps and the use of them for super-resolution. Bicubic interpolation is used as a baseline super-resolution method for quality comparison. The accelerated solver is implemented using the OpenCL 1.2 framework and deployed on two GPU platforms: GeForce GTX 1080 Ti GPU and GeForce GTX 680. The same algorithm is implemented and

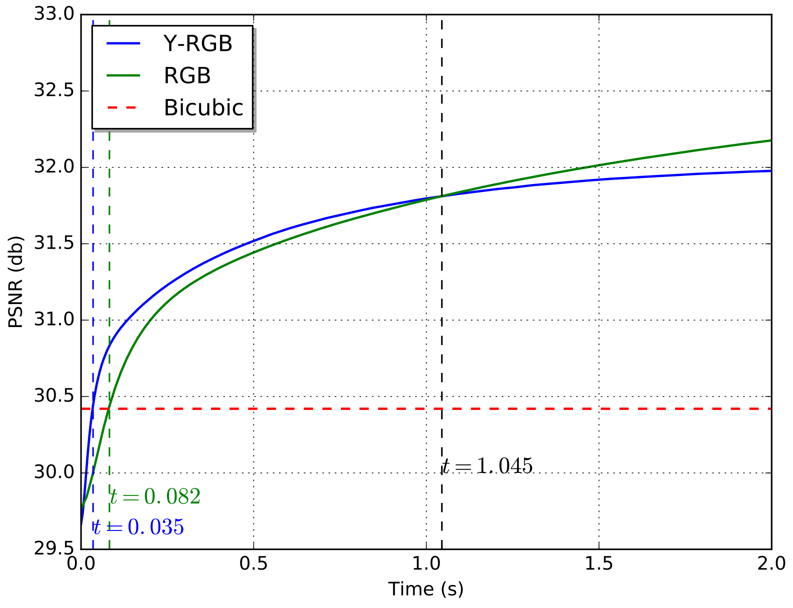


Figure 4.15: Rate distortion curve of *Y-RGB* and *RGB* strategies running on GeForce GTX 1080 Ti

executed on CPU devices to measure the speeding up of the proposed approach. In this experiment, parameters are set as follow, $\lambda = 0.7$, $\alpha = 0.04$, $|\mathcal{N}_w(\mathbf{u})| = 1$ (Eq. 4.9). The number of iterations is varied to see the changes in output quality which is measured by the Peak Signal to Noise Ratio (PSNR).

The *'dino'* 4D light field data has the resolution of $9 \times 9 \times 512 \times 512$ in which 9×9 denotes the directional resolution (81 viewpoints in total) and 512×512 denotes the spatial resolution (or the resolution of each sub-aperture image). In order to prepare the low-resolution 4D light field for our experiment, the original 4D light field is spatially downsampled by a factor of 4. Therefore, the input 4D light field has the resolution of $9 \times 9 \times 128 \times 128$. The disparity estimation algorithm is applied to this input data to extract a set of disparity maps associated with each of the viewpoints (81 in total). These disparity maps are then upsampled by a factor of 4 so that they can be used in our warping kernel. For each low-resolution sub-aperture image, the extra 4 direct neighbors of it are employed as the input for the super-resolution algorithm. For example, let $\boldsymbol{\theta}_0 = \begin{bmatrix} 4 & 4 \end{bmatrix}^T$ is the directional index of the low-resolution image of which the high-resolution image is going to be estimated, this image along with the other 4 images at directional indexes $\left\{ \begin{bmatrix} 3 & 4 \end{bmatrix}^T, \begin{bmatrix} 5 & 4 \end{bmatrix}^T, \begin{bmatrix} 4 & 3 \end{bmatrix}^T, \begin{bmatrix} 4 & 5 \end{bmatrix}^T \right\}$ will become the input for the super-resolution algorithm. In total, there are 5 low-resolution images and 5 disparity maps associated with them are used for super-resolution. Without using the ground-truth disparity map, we observe that this is the best number of input images which gives a good

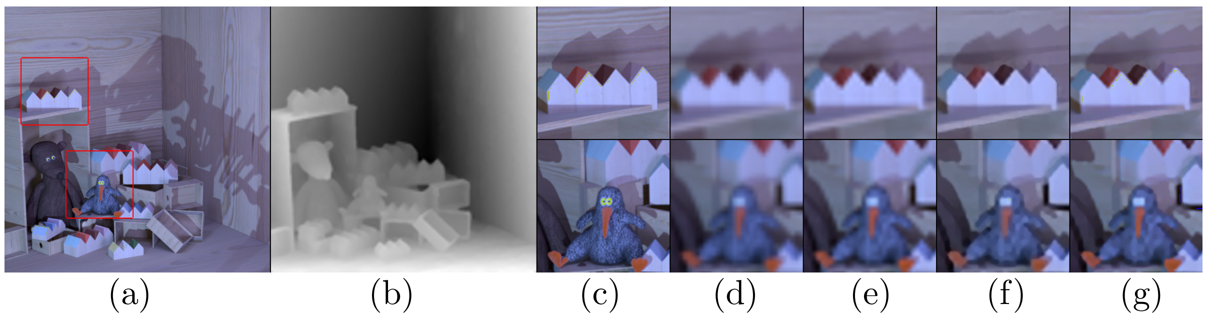


Figure 4.16: Light field image 4× super-resolution results. (a) Ground-truth high-resolution sub-aperture image with two region of interests marked with red rectangles. (b) Estimated disparity map. (c-g) are the zoom-in images of the two marked regions on the ground-truth image, high-resolution images generated by bilinear, bicubic, *Y-RGB* and *RGB* approach, respectively

quality high-resolution image. More input images will result in worse quality due to the error introduced by estimated disparity maps.

For RGB color image super-resolution, two strategies are applied. The first strategy is transforming the RGB input light field to YCbCr color space and applying super-resolution algorithm only on Y channel. This strategy is motivated by the fact that Cb and Cr channels are responsible for chrominance information which contributes less to the sharpness of a color image. By applying the super-resolution on only a single color channel, further speeding up can be achieved with an acceptable reduction in the quality of a high-resolution image. In this strategy, the RGB 4D light field is first transformed to YCbCr color space. The Y channel images are extracted and super-resoluted using the proposed GPU-based architecture. The Cb and Cr channel images are simply upsampled by using bilinear interpolation. The super-resoluted Y channel images are then combined with Cb and Cr channel images and transformed back to RGB color space. The second strategy is applying super-resolution algorithm on 3 color channels (i.e. R, G, B channels) separately. This approach requires a higher computation time compared to the previous strategy but provides a higher quality output. The first strategy and the second strategy are labeled as *'Y-RGB'* and *'RGB'* respectively.

Fig. 4.15 compared the output quality of the two strategies in term of runtime and psnr value. The psnr of the upsampled high-resolution image using bicubic interpolation is 30.42. Our GPU-based implementation of *Y-RGB* and *RGB* can quickly pass this baseline result at 0.035s and 0.082s respectively. As expected, the *Y-RGB* strategy requires less time for computation and scores better for high-speed constraint (less computation time) compared to *RGB* strategy. However, *RGB* can achieve better quality output if enough computation time is given. The *RGB* strategy begins to get higher psnr than *Y-RGB* after 1.045s and continue increasing.

Table 4.3: The computation time comparison of 4D light field super-resolution on different platforms. The time is measured for a single gradient step.

Strategy	**Platform**	**Time (ms)**	**Speed-up**
Y-RGB	2.2Ghz Intel Core i5	993	1.0
Y-RGB	NIVIDA Geforce GTX 680	9.17	108.28
	NIVIDA Geforce GTX 1080i	4.87	203.90
RGB	NIVIDA Geforce GTX 680	28.17	35.25
	NIVIDA Geforce GTX 1080i	13.94	71.23

Table 4.3 reports the computation time of super-resolution algorithm running on different platforms. The computation time is measured for a single steepest gradient descent step. Compared to *Y-RGB* strategy implemented for Intel CPU, the implementation of the same strategy executed on GTX 680 and GTX 1080i can achieve the speeding up by the factor of $108\times$ and $203\times$ respectively. *RGB* strategy generally requires $3\times$ more time than *Y-RGB*, but it is still faster than the CPU-based implementation of *Y-RGB* by $35\times$ and $71\times$ when executed on GTX 680 and GTX 1080i respectively. It is also noticeable that the execution time of GTX 1080i is almost $2\times$ shorter than that of GTX 680. This is reasonable since the first GPU device is more powerful than the second. GTX 1080i possesses $2.3\times$ more computation units (CUDA cores) and $1.5\times$ faster clock frequency than GTX 680.

Fig. 4.16 compared the high-resolution images calculated by the GPU-based implementation of proposed strategies (*Y-RGB* and *RGB*) and the two baseline methods: bilinear and bicubic interpolations. It can be seen from the figure that our estimated high-resolution image possesses higher visual quality than the interpolation methods. Looking closely at the two zoom-in regions, the proposed approach provides sharper images with more details compared to bicubic and bilinear interpolations. Comparing between *Y-RGB* and *RGB* strategies, the *RGB* strategy provides even a better visual quality, especially in regions where color changes are high.

4.5.3 Performance Analysis of GPU-based ADMM solvers

For analyzing the performance improvement of the proposed GPU accelerated approach, we perform an evaluation of three realization strategies of ADMM iterative solvers. Fig. 4.18 reports the cumulative execution time of the three GPU implementations. The initial GPU implementation (i.e., `buf`) is considered as a baseline, in which a 1D buffer object is used

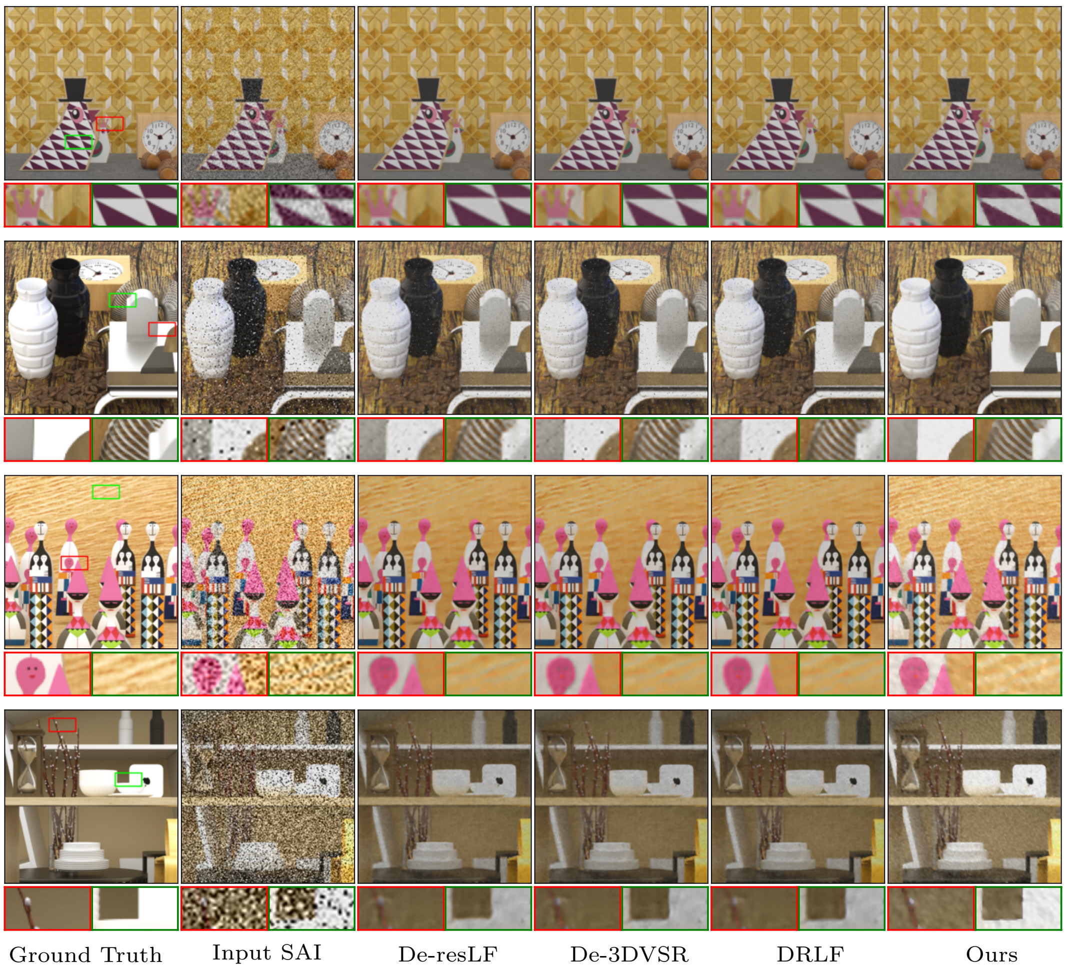


Figure 4.17: Visual comparisons of LFSR approaches under various mixed noise settings. From top to bottom ('scene' - σ/ν): 'Rooster-clock' - 20/0; 'Coffee-beans-vases' - 20/5; 'Smiling-crowd' - 50/0; 'Dishes' - 50/20.

for holding variable and input data in GPU global memory. In the second implementation, denoted as *i2d*, 1D buffer objects are replaced by Image2D objects. This allows us to make use of the texture cache provided in GPU architecture for speeding up access to image-like data. The third implementation, denoted as *i2d_local* takes advantage of local memory for buffering and sharing data within a work-group. Since local memory is close to the computing unit, this provides a high-speed data pool for kernel tasks which frequently require access to multiple neighbor pixels (i.e. blurring, warping). For this experiment, we use 5×5 angular views as input for ×4 SR to a spatial resolution of 512×512. The number of ADMM iterations and CG iterations is set to 10 and 5, respectively. The execution time of the ADMM solver can be divided into three parts. The *io* part covers the time for transferring input data from CPU memory into GPU global memory and reading back the reconstructed HR image from GPU to GPU memory. The $wz - step$ part represents the computation time

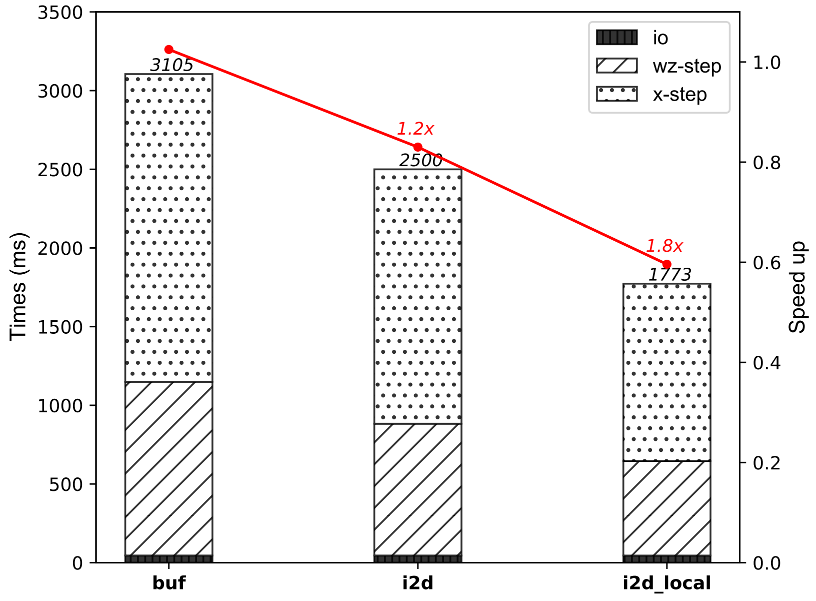


Figure 4.18: Cumulative execution time and speed-up after each optimization

of updating **w** and **z** in an ADMM iteration, while $x - step$ part measures the time to solve for **x** by applying conjugate gradient descent technique, see Fig. 4.6. As could be seen from Fig. 4.18, the IO time only accounts for a small amount of overall execution time, while most of the time is spent on $x - step$ and $wz - step$. As compared to the buf version, the texture cache provided by Image2D object *i2d* does shorten the computation time of ADMM solver by a factor of 1.2×. The local memory sharing technique further speeds up the computation time by a factor of 1.8×.

The advantage of using the OpenCL framework is that the accelerated solver can be executed on various platforms. Fig. 4.19 shows the execution time of *i2d_local* on various GPU platforms. In this test, we vary the number of input LR images: 9, 25, 49, and 81, which are denoted as 3×3,5×5,7×7, and 9×9, respectively. The regularization window size is configured to 5×5, and the number of conjugate gradient steps is set to 5. For each case, we measure the execution time of a single ADMM iteration and compare it to the CPU implementation, executed on i7-5820K 3.30GHz. In general, we observed a higher speed up as compared to CPU execution when more input images are provided. The speedup ranges from 21× to 42× in the case of 3×3 inputs and from 32× to 75× in the case of 9×9 inputs.

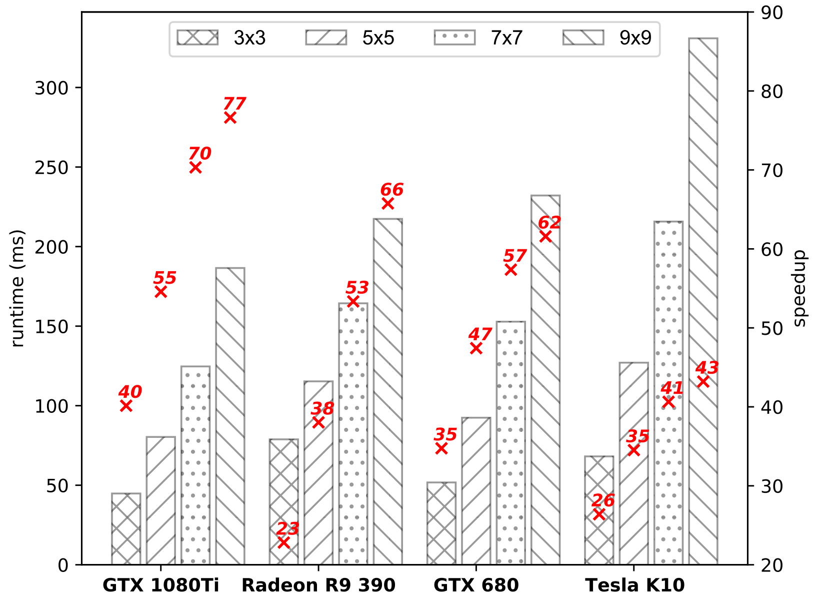


Figure 4.19: Execution time of GPU-based ADMM solver under different number of inputs on various OpenCL platforms

4.5.4 Comparison to LFSR Approaches

In this section, we evaluate the performance of the proposed method under severe mixed noise conditions and compare it to the related approaches (i.e., resLF [61], DRLF [118], and 3DVSR [27]). These approaches currently provide state-of-the-art performance in reconstructing high-resolution LF images. To the best of our knowledge, only DRLF [118] supports LFSR with noisy input. For the evaluation, we randomly select five scenes from the Inria LF dataset [15]. For each scene we generate low resolution LF ($\times$2) and insert noises with four configurations, (σ=20, ν=0%), (σ=20, ν=5%), (σ=50, ν=0%), (σ=50, ν=20%). These Gaussian noise settings are selected due to the pre-trained weights published by DRLF. DRLF needs different trainings for dealing with different noise conditions, and there are only three pre-trained weights published for three Gaussian noise configurations σ=10, σ=20, and σ=50. In addition, DRLF does not directly process noisy LR inputs. It provides separate networks for de-nosing and super-resolution. Therefore, we applied first their de-noising network to noisy LR inputs and then applied their SR network to the de-noised LR outputs. In this way, we are able to evaluate the performance of the other two state-of-the-art LFSR approaches (i.e., resLF [61], 3DVSR [27]) using the de-noised LR output from DRLF.

The experimental results are reported in Table 4.4 and visualized in Fig. 4.17. For the two approaches, resLF and 3DVSR, which do not support noisy LF input, we generate de-noised

LF with DRLF and use it as an input to resLF and 3DVSR. These results are denoted as *De+resLF* and *De+3DVSR* respectively. For all noise settings, our approach provides the best reconstruction quality in terms of PSNR. For mixed noise settings, the proposed method achieves an averagely highest SSIM score as compared to the other approaches. These high scores pay tribute to the robustness of the proposed model in which de-noising and super-resolution are jointly resolved. Without de-nosing resLF and 3DVSR completely fails to reconstruct a good quality HR image. In practice, they up-scale not only the texture but also the existing noise. Their scores are, therefore, even worse as compared to bi-cubic upsampling approach in which noises are blurred out. From Fig. 4.17, it is evident that the reconstructed HR image from the other approaches is over-smoothed while our approach preserves well the texture content and high-frequency information, e.g., and object edges in *Dishes* scene, background pattern in *Smiling-crowd* scene. Since DRLF supports only Gaussian noise, it fails to recognize impulse noise in the LR input. The impulse noise is either ignored, i.e., when Gaussian noise level is low, or mistreated, i.e., in a severe Gaussian noise setting. Consequently, the reconstructed HR images are presented with noisy traces, i.e., *Coffee-beans-vases* scene or losing texture detail, i.e., the flower bud in

Table 4.4: Quantitative comparison of LFSR approaches under various mixed noise settings.

Noise (σ/ν)	Scenes	BIC (psnr/ssim)	resLF (psnr/ssim)	De+resLF (psnr/ssim)	3DVSR (psnr/ssim)	De+3DVSR (psnr/ssim)	DRLF (psnr/ssim)	Ours (psnr/ssim)
20/0	Dishes	22.63/0.378	21.20/0.316	28.80/0.886	19.32/0.252	28.74/0.891	28.71/0.888	30.47/0.846
	Rooster-clock	22.81/0.375	21.24/0.304	30.73/0.864	19.36/0.241	31.15/0.879	30.84/0.881	31.48/0.798
	Coffee-beans-vases	21.55/0.507	20.41/0.453	25.24/0.801	18.86/0.389	25.51/0.812	25.62/0.818	26.31/0.801
	Smiling-crowd	21.81/0.493	20.56/0.428	25.93/0.836	18.84/0.359	26.11/0.853	25.97/0.851	29.27/0.832
	Electro-devices	22.67/0.322	21.17/0.260	28.41/0.824	19.28/0.201	28.67/0.836	28.26/0.823	30.74/0.793
	mean	22.29/0.415	20.91/0.352	27.82/0.842	19.13/0.288	28.04/**0.854**	27.88/0.852	**29.66**/0.814
20/5	Dishes	18.89/0.283	17.51/0.234	26.15/0.671	15.31/0.177	25.31/0.629	26.18/0.663	30.36/0.843
	Rooster-clock	19.19/0.267	17.69/0.212	27.91/0.710	15.42/0.158	27.18/0.680	27.80/0.720	31.39/0.794
	Coffee-beans-vases	18.25/0.394	16.99/0.342	23.56/0.646	14.96/0.274	23.13/0.625	23.81/0.652	26.22/0.797
	Smiling-crowd	18.14/0.383	16.86/0.329	23.97/0.693	14.53/0.249	23.17/0.658	23.86/0.695	29.12/0.829
	Electro-devices	18.93/0.231	17.51/0.184	26.00/0.631	15.30/0.135	25.29/0.595	25.76/0.621	30.65/0.789
	mean	18.68/0.312	17.31/0.260	25.52/0.670	15.10/0.199	24.82/0.638	25.48/0.670	**29.55/0.810**
50/0	Dishes	16.11/0.162	14.56/0.129	20.81/0.819	11.81/0.086	20.51/0.818	20.62/0.821	27.74/0.715
	Rooster-clock	15.90/0.133	14.28/0.101	21.62/0.787	11.43/0.065	21.36/0.789	21.28/0.793	28.55/0.669
	Coffee-beans-vases	15.92/0.246	14.44/0.196	19.85/0.693	11.83/0.138	19.69/0.694	19.71/0.709	24.26/0.672
	Smiling-crowd	16.28/0.247	14.80/0.203	19.02/0.686	12.13/0.146	18.88/0.695	18.91/0.692	26.40/0.717
	Electro-devices	15.99/0.120	14.41/0.093	21.10/0.743	11.60/0.062	20.92/0.745	20.81/0.732	28.20/0.662
	mean	16.04/0.182	14.50/0.145	20.48/0.746	11.76/0.100	20.27/0.748	20.27/**0.749**	**27.03**/0.687
50/20	Dishes	13.00/0.094	11.78/0.075	18.87/0.743	9.50/0.047	18.66/0.736	18.85/0.745	26.89/0.704
	Rooster-clock	13.23/0.080	11.91/0.061	19.76/0.721	9.48/0.038	19.56/0.714	19.40/0.721	27.80/0.675
	Coffee-beans-vases	12.78/0.145	11.61/0.115	17.82/0.622	9.45/0.077	17.69/0.618	17.60/0.635	23.78/0.674
	Smiling-crowd	12.66/0.148	11.52/0.121	16.44/0.621	9.44/0.083	16.33/0.622	16.39/0.625	25.54/0.721
	Electro-devices	13.05/0.068	11.78/0.053	19.40/0.690	9.45/0.033	19.25/0.684	19.09/0.677	27.44/0.656
	mean	12.94/0.107	11.72/0.085	18.46/0.680	9.46/0.056	18.30/0.675	18.26/0.681	**26.29/0.686**

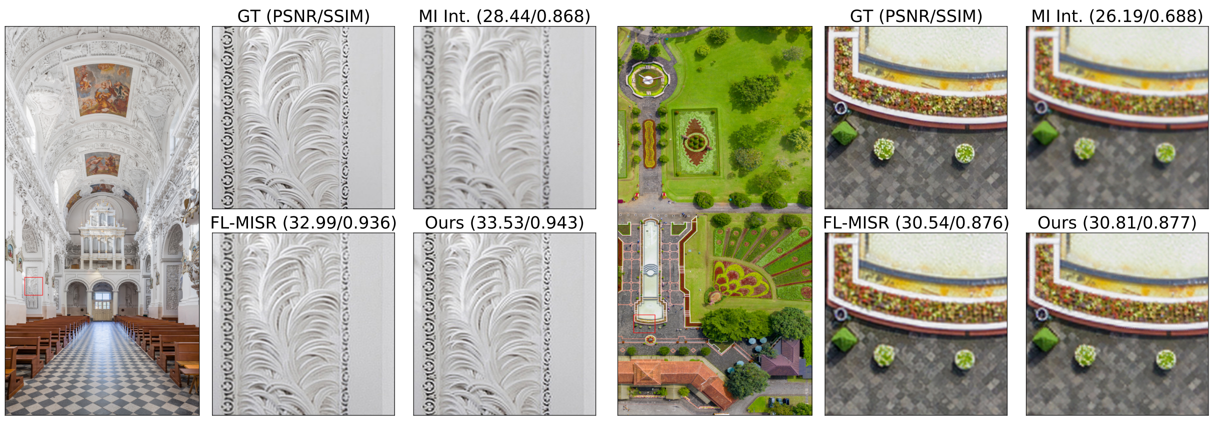


Figure 4.20: HR reconstruction results of DIV8K dataset [98]. *left* ×2 results of image 0002; *right* ×4 results of image 0084.

Dishes scene.

4.5.5 Comparison to GPU-Accelerated Approach

Table 4.5: Evaluation of parallel computing approach for MISR problem on 8-bit natural images in DIV8K dataset. MI Int.: Multi-image interpolation (56 cores Intel Xeon Gold 5120), FL-MISR [64] (4 GTX 1080i), ours (1 GTX 1080i).

Image Index		0001	0002	0007	0027	0055	0066	0084
Resolution of GT		5376×5760	5568×5760	1920×2880	2112×2880	5760×5760	1920×2880	5760×3840
Upscaling 2×								
MI Int.	PSNR/SSIM	30.49/0.9215	28.44/0.8677	33.68/0.8810	28.37/0.8988	33.80/0.9018	35.21/0.9296	29.11/0.8277
	Runtime (*s*)	0.51	0.52	0.11	0.20	0.53	0.11	0.36
FL-MISR	PSNR/SSIM	37.11/0.9620	32.99/0.9360	35.09/0.9111	33.21/0.9417	38.03/0.9564	37.12/0.9452	34.13/0.9410
	Runtime (*s*)	1.50	1.29	0.69	0.71	1.3	0.66	1.21
Ours	PSNR/SSIM	37.24/0.9713	33.53/0.9430	35.42/0.9220	33.73/0.9497	38.13/0.9616	37.51/0.9539	34.60/0.9454
	Runtime (*s*)	0.92	1.14	0.15	0.24	0.72	0.18	0.83
Upscaling 3×								
MI Int.	PSNR/SSIM	26.74/0.8460	25.65/0.7749	32.03/0.8395	25.15/0.8212	30.79/0.8153	32.65/0.8968	26.19/0.6883
	Runtime (*s*)	1.00	0.99	0.11	0.13	0.55	0.11	0.38
FL-MISR	PSNR/SSIM	33.24/0.9446	29.43/0.8941	33.99/0.8941	30.17/0.9139	35.90/0.9379	36.06/0.9398	30.54/0.8764
	Runtime (*s*)	1.78	1.73	0.32	0.38	1.93	0.35	1.65
Ours	PSNR/SSIM	33.39/0.9517	30.04/0.9017	34.50/0.8984	30.57/0.9293	36.22/0.9402	36.35/0.9447	30.81/0.8774
	Runtime (*s*)	1.13	1.17	0.23	0.25	1.22	0.23	0.84

As discussed in Sec. 4.1, the proposed framework shares a similar setup as a multi-frame super-resolution problem and indeed can be applied as well for this kind of problem. To evaluate the performance of our accelerated framework, we conducted an experiment on the natural image dataset DIV8K [98] and compared to recent related work on the field (FL-MISR [64]). We follow the experimental setup described in [64] to prepare the low-resolution images and perform the HR image reconstruction with our accelerated solver. Particularly, we pick up seven images from DIV8K dataset and generate, for each of them, four LR images for ×2 SR and nine LR images for ×3 SR. The shifting of ×2 and ×3 image sets are respectively $\frac{1}{2}$px and $\frac{1}{3}$px. Since FL-MISR use ℓ^1 data fidelity and BTV regularization in their model, we turn off our ℓ^2 term and configure nonlocal weighting (i.e. $W_{\mathbf{d}}$) to match BTV condition. The accelerated ADMM iterative solver is then executed to minimize the cost function in Eq. 4.14. For a fair comparison, we stop our iterative solver as soon as the quality of the reconstructed image is comparable to FL-MISR and measure the execution time. Quantitative evaluation results are listed in Table 4.5, while visual comparison are shown in Fig. 4.20. From the table, it is obvious that our GPU accelerated solver outperforms FL-MISR in processing speed for all test cases while providing a better reconstruction quality. As compared to FL-MISR, our GPU-based solver achieves an average speed-up of 2.46× and 1.57× for up-scaling ×2 and up-scaling ×3 respectively. This performance boost tributes to the effectiveness of ADMM solver and the realization strategy of transformation matrices $(A_k, S_{\mathbf{d}})$. In contrast to FL-MISR, which chooses to implement $A_k, S_{\mathbf{d}}$ with sparse matrices, our approach takes advantage of linear functions (i.e., $\mathcal{W}, \mathcal{B}, \mathcal{D}$) to optimize GPU memory and computation resource. Therefore, our GPU-based solver can fit well within a single GTX 1080i GPU, while FL-MISR needs four of them to solve the same problem.

4.6 Summary

This chapter presents a GPU-accelerated computational framework for reconstructing high-resolution perspective image from 4DLF data under mixed Gaussian-Impulse noise conditions. The proposed SR model derived from a statistical perspective takes advantage of a joint $\ell^1 - \ell^2$ data fidelity term for dealing with mixed noise conditions and weighted non-local total variation for enforcing LF image prior. Our approach combines the de-noising effect and SR reconstruction into a single optimization problem which, as shown in the experimental results, allows us to surpass the current state-of-the-art approaches in which de-noise and SR problems are resolved separately. The non-smooth convex optimization problem resulting from the proposed SR model is effectively solved by ADMM algorithm. By transforming the minimization of $\ell^1 - \ell^2 - \ell^1$ mixture cost function into least square approximation and proximal operator problems, ADMM overcomes the main problem of gradient descent technique in finding a suitable step-size. We showed that GPU acceleration is well-suited to speeding up the iteratively solving process. To verify the robustness of the proposed SR model and evaluate the performance of the accelerated optimizer, an extensive experiment is conducted on 4D synthetic LF dataset and high-resolution natural image dataset. The experimental results show that the proposed approach outperforms the previous work in accelerating the super-resolution task and optimizing GPU resources. While providing a better reconstruction quality, our accelerated framework provides an average speed up of 2.46× and 1.57× for ×2 and ×3 SR tasks, respectively. The accelerated solver achieves a speedup of 77× as compared to CPU implementation.

Chapter 5

EPI Volume-based High-Resolution Light Field Reconstruction

This chapter presents a deep learning-based approach for reconstructing high resolution 4DLF. Its main contribution is threefold. First, a novel 3D EPI volume-based framework for addressing various LF SR problems, i.e., ASR, SSR, and ASSR are developed. Specifically, the framework comprises two consecutive stages, i.e., preliminary up-sampling stage and volume-based enhancement stage. The first stage allows different options to be used for up-sampling the input volume to the desired resolution. Then, in the second stage, the novel EVRN corrects the high-frequency information by incorporating both spatial information in SAI and angular information in 2D EPI to reconstruct a high-quality LF image. Secondly, a simple but effective deep CNN model is proposed for preliminary up-sampling angular dimensions. Although its output quality is later greatly improved by EVRN, this model itself already surpasses existing approaches in ASR. Thirdly, an extensive evaluation is conducted on 90 challenging synthetic and real-world LF scenes from 7 public LF datasets. In this evaluation, we analyzed the performance of the proposed approach and compared it to state-of-the-art approaches. Parts of the results of this chapter have been published in [27].

5.1 Problem Statement and Notation

For a better observation and analysis, various ways to visualize 4D LF are introduced in the literature. Common visualizations are using 2D slices, i.e., sub-aperture image (SAI), angular patch image (API), or epipolar image (EPI). By fixing the directional index $\boldsymbol{\theta} = \boldsymbol{\theta}_i$ or the spatial index $\mathbf{z} = \mathbf{z}_i$, one can respectively obtain a SAI $\boldsymbol{L}(\mathbf{z}, \boldsymbol{\theta}_i)$ or an API $\boldsymbol{L}(\mathbf{z}_i, \boldsymbol{\theta})$. Fig. 1.1 (b),(c) are examples of a sub-aperture image and an angular patch extracted from a 4D synthetic light field image, *dino* [4]. The resolution of this light field is 512×512×9×9, where 512×512 is the spatial resolution and 9×9 is the angular resolution. The angular patch consists of pixels, each at the same spatial location $\mathbf{z}_i = (78, 170)^T$ on a sub-aperture. An EPI can be acquired by fixing one index in the spatial plane and one index in the angular plane while varying the remaining two indices. By fixing vertical indices, i.e., $y = y_i, \tau = \tau_i$, we have the horizontal EPI $\Sigma_{y_i,\tau_i}(x, \rho) = \boldsymbol{L}\big([x, y_i]^T, [\rho, \tau_i)]^T\big)$. A similar procedure applies to vertical EPI $\Sigma_{x_i,\rho_i}(y, \tau)$. Fig. 5.1 illustrates the two types of EPIs.

As pointed out in [24], a point in 3D space is projected onto a line in EPI whose slope is decided by the depth of this point. For example, in Figure 5.1 the point located at (x_i, y_i) projected onto two lines in horizontal EPI (Σ_{y_i,τ_i}) and vertical EPI (Σ_{x_i,ρ_i}). Notice that all letters crossed by the line $y = y_i$ (i.e., dotted green line) reside on the same depth and result in identical slope in horizontal EPI Σ_{y_i,τ_i}. This property of EPI represents the geometric information of the scene and was exploited in many LF image processing applications (i.e. disparity estimation [24, 26], super-resolution [24, 57]). In this work, we consider a 3D version of EPI, namely EPI volume [26], in which the second spatial dimension is added. The intuition behind this volume data is the combination of both spatial coherence (i.e. x vs. y) and EPI coherence (i.e. x vs. ρ) that can be employed for reconstructing high-quality LF image. Horizontal EPI volume $\boldsymbol{V}_{\tau_i}$ and vertical EPI volume $\boldsymbol{V}_{\rho_i}$ are defined in Equation 5.1. An EPI volume is an orthogonal 3D slide through 4D LF and can be considered as a stack of 2D EPI along a spatial axis, e.g., $\boldsymbol{V}_{\tau_i}$ is constructed by stacking horizontal EPI Σ_{y_i,τ_i} along spatially vertical axis y.

$$
\begin{aligned}
\boldsymbol{V}_{\tau_i}: \mathbb{R}^3 \rightarrow \mathbb{R}, \qquad (x, \rho, y) \rightarrow \boldsymbol{L}\left(\begin{bmatrix} x \\ y \end{bmatrix}, \begin{bmatrix} \rho \\ \tau_i \end{bmatrix}\right) \\
\boldsymbol{V}_{\rho_i}: \mathbb{R}^3 \rightarrow \mathbb{R}, \qquad (y, \tau, x) \rightarrow \boldsymbol{L}\left(\begin{bmatrix} x \\ y \end{bmatrix}, \begin{bmatrix} \rho_i \\ \tau \end{bmatrix}\right)
\end{aligned}
\tag{5.1}
$$

Let us define that 4D LF image has the resolution of $W{\times}H{\times}A{\times}A$, where $H, W \in \mathbb{N}^+$ respectively represent the height and width of each SAI and $A = 2K + 1, K \in \mathbb{N}^+$ denotes the angular resolution. The resolutions of ρ-axis and τ-axis are set equally, since this square array of views is commonly used in literature [52, 54, 57, 59–61] and available in public dataset [4, 14–19]. The resolution of horizontal EPI volume $\boldsymbol{V}_{\tau_i}$ and vertical EPI volume $\boldsymbol{V}_{\rho_i}$ are then $W{\times}A{\times}H$ and $H{\times}A{\times}W$ respectively.

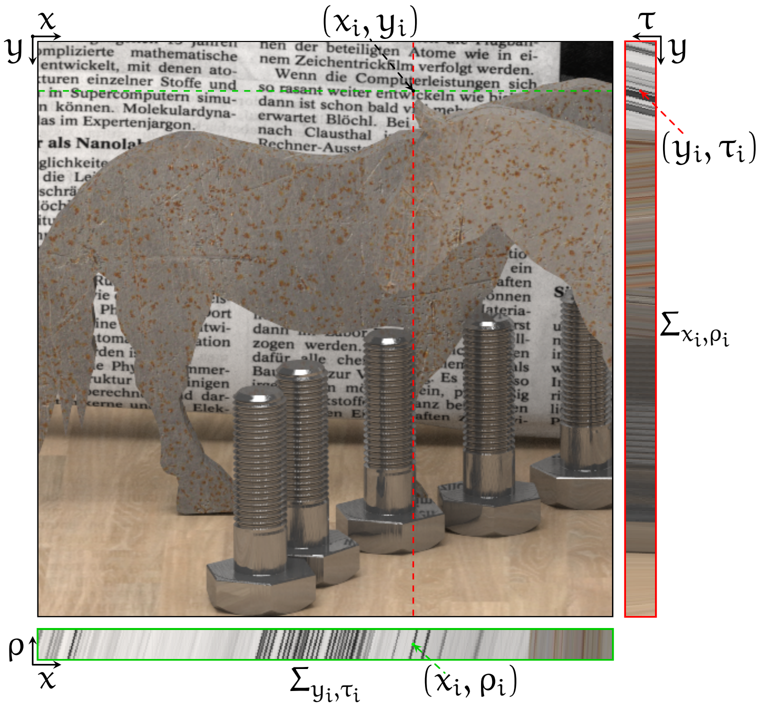


Figure 5.1: 2D EPI projection of 4D LF scene *horses* [14]. *top-left* SAI at $\boldsymbol{\theta}_i = [\rho_i, \tau_i]^T$; *top-right* vertical EPI (Σ_{x_i,ρ_i}); *bottom* horizontal EPI (Σ_{y_i,τ_i}).

5.2 EPI Volume-based LF Super-Resolution

5.2.1 Overview of the Proposed Approach

The proposed approach targets the super-resolution of 3D projected versions of a 4D LF image. Instead of directly upsampling a 4D LF image (e.g, SR4D [60]), we first decompose it into a complete set of 3D EPI volumes. Each EPI volume is then super-resolved to its desired resolution. The high-resolution sets of EPI volumes are finally merged to form the final high-resolution 4D output. Fig. 5.2 illustrates the overall process of the proposed EPI volume SR framework. The framework is comprised of two main processing stages: a preliminary up-sampling stage and a volume enhancement stage. The first stage involves two tasks: preliminary spatial super-resolution (PSSR) and preliminary angular super-resolution (PASR). The second stage includes an EPI volume-based refinement network (EVRN). Given a 3D EPI volume ($\boldsymbol{V}_l$) with the resolution of $W{\times}A{\times}H$, PSSR and PASR sequentially up-sample the spatial resolution to $\zeta_{\mathbf{z}}W{\times}\zeta_{\mathbf{z}}H$ and the angular resolution to $\zeta_{\theta}A$, where $\zeta_{\mathbf{z}}$ and ζ_{θ} are spatial and angular scaling factor respectively. This preliminarily up-sampling volume is provided as an input to EVRN which will, in turn, refine the 3D EPI structure

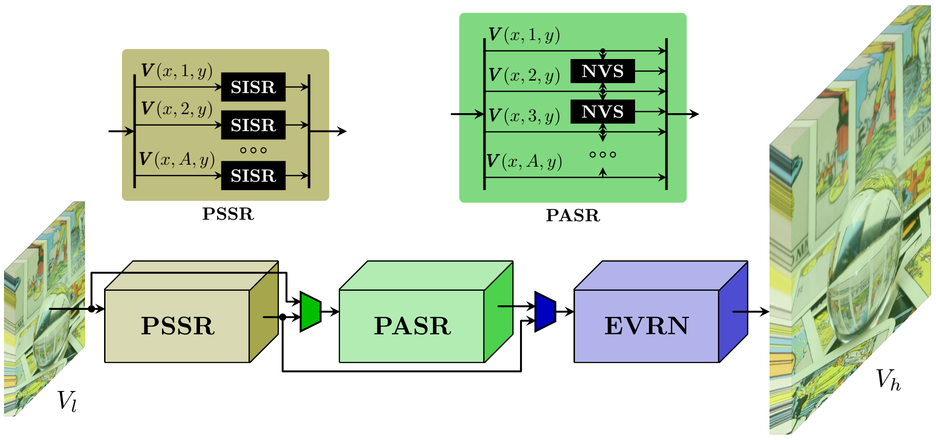


Figure 5.2: Overview of 3D EPI volume SR framework. The framework consists of a preliminary upsampling stage and an enhancement stage. The earlier stage includes a preliminary spatial SR block (PSSR) and a preliminary angular SR block (PASR). The later stage includes an EPI volume refinement network (EVRN).

and return an enhanced high-resolution volume. PSSR and PASR can be applied separately or jointly depending on SR applications (i.e., SSR, ASR, and ASSR).

Algorithm 6: Volume-based super-resolution (VSR) function

```
1 Function VSR($L_l$, $\varepsilon$, $\mathcal{F}$):
2     $\{V\}_\varepsilon$ := Slice($L_l$, $\varepsilon$)          ▷ Extract $\varepsilon$-axis volumes
3     $\{V\}_*$ := {}                                ▷ Initialize empty volume set
4     for $V$ in $\{V\}_\varepsilon$ do
5         $\{V\}_* := \{V\}_* + \mathcal{F}(V)$
6     end
7     $L_h$ := Merge($\{V\}_*$, $\varepsilon$)                  ▷ Reconstruct 4D LF
8     return $L_h$
9 end
```

The function VSR in Algorithm 6 describes the application of the proposed framework for the reconstruction of high-resolution LF image. VSR takes in three parameters: a low-resolution LF ($\boldsymbol{L}_l$), a directional axis ($\varepsilon \in \{\rho, \tau\}$), and a volume-based SR function ($\mathcal{F}$). The directional axis is needed to determine the direction of 3D projection which will result in either vertical volume ($\varepsilon = \rho$) or horizontal volume ($\varepsilon = \tau$). The volume-based SR function encodes the configuration of PSSR and PASR as in Fig. 5.2, i.e., only PSSR, only PASR, and both. The procedure of VSR is as follows. First, a set of EPI volumes are extracted from the input $\boldsymbol{L}_l$ (line 2). Depending on ε, function Slice($\cdot$) will return either horizontal volume or vertical volume set $\big(\{\boldsymbol{V}\}_\varepsilon = \{\boldsymbol{V}_{\varepsilon_i}, i = 1, .., A\}\big)$. Next, a high-resolution volume set $\{\boldsymbol{V}\}_*$ is acquired by applying volume SR function ($\mathcal{F}$) to each volume in $\{\boldsymbol{V}\}_\varepsilon$ (line

3 − 6). Finally, we combine 3D volumes in $\{\boldsymbol{V}\}_*$ to form a high-resolution 4D output (line 7).

Algorithm 7: Spatial-angular super-resolution of 4D LF

```
Input: $\boldsymbol{L}_l$
Output: $\boldsymbol{L}_h$
1  $\boldsymbol{L} := \boldsymbol{L}_l$
2  if spatial super-resolution then
3      $\mathcal{F} := \mathcal{R}(\mathcal{S}(\cdot))$                          ▷ Apply PSSR and EVRN
4      $\boldsymbol{L} := 0.5\Big(\mathsf{VSR}(\boldsymbol{L}, \tau, \mathcal{F}) + \mathsf{VSR}(\boldsymbol{L}, \rho, \mathcal{F})\Big)$
5  else if angular super-resolution then
6      if spatial super-resolution then
7          $\mathcal{F} := \mathcal{S}(\cdot)$                              ▷ Apply only PSSR
8          $\boldsymbol{L} := \mathsf{VSR}(\boldsymbol{L}, \tau, \mathcal{F})$
9      end
10     $\mathcal{F} := \mathcal{R}(\mathcal{A}(\cdot))$                          ▷ Apply PASR and EVRN
11     $\boldsymbol{L} := \mathsf{VSR}(\boldsymbol{L}, \tau, \mathcal{F})$                       ▷ Upscale $\rho$-axis
12     $\boldsymbol{L} := \mathsf{VSR}(\boldsymbol{L}, \rho, \mathcal{F})$                       ▷ Upscale $\tau$-axis
13 end
14 $\boldsymbol{L}_h := \boldsymbol{L}$
```

The applications of VSR function in SSR, ASR, and ASSR are described in Algorithm 7. In the case of SSR, $\mathcal{F}$ is comprised of PSSR, denoted as $\mathcal{S}$, and EVRN denoted as $\mathcal{R}$, (line 3). VSR is applied to horizontal and vertical volumes separately and then the output LFs are averaged (line 4). In the case of ASR, $\mathcal{F}$ is set as PASR, denoted as $\mathcal{A}$, followed by EVRN (line 10). The super-resolution of horizontal volumes (line 11) and vertical volumes (line 12) will increase the angular resolution to $\zeta_\theta A \times A$ and $\zeta_\theta A \times \zeta_\theta A$ respectively. As for ASSR, a similar procedure to ASR is applied, except that PSSR is previously employed to spatially up-sample the input light field (line 7,8).

5.2.2 EPI Volume Refinement Network

The proposed network bases on global residual learning architecture that is implemented with a long skip connection and an element-wise addition as illustrated in Fig. 5.3. Global residual learning allows EVRN to avoid learning complicated transformation and focus on the reconstruction of high-frequency information differing between low and high-resolution EPI volume. 3D convolutional kernels are utilized in our network instead of 2D convolutions since this type of kernel was shown to be effective with 3D EPI volume data [26]. EVRN is comprised of two parts: attention-based residual learning extracts densely residual-based

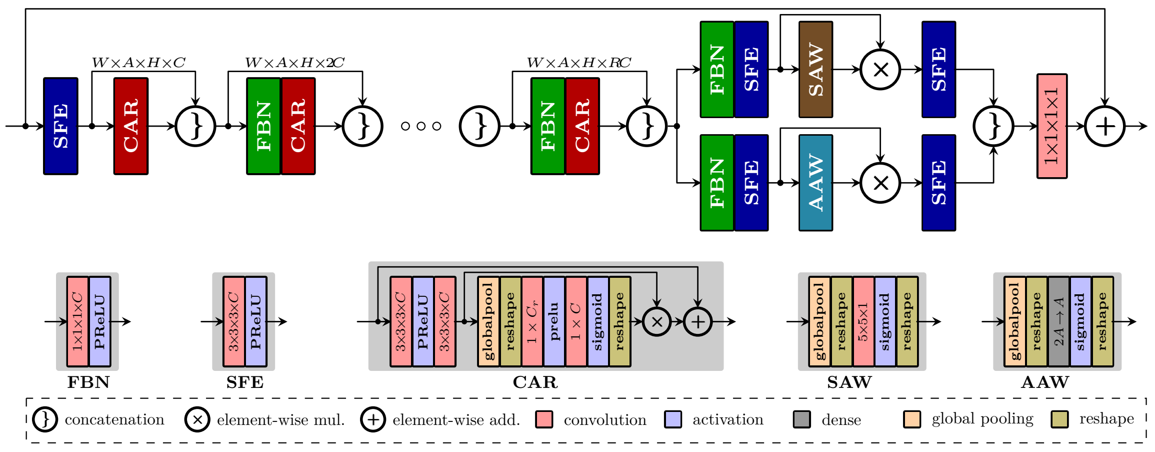


Figure 5.3: The network architecture of the proposed EPI volume refinement network (EVRN)

features and attention-based multi-path learning reconstructs high-frequency information.

5.2.2.1 Attention-based Residual Learning

This part consists of a shallow feature extraction layer (SFE) followed by R local residual learning blocks. Dense connection [119] is employed to alleviate gradient vanishing and improve signal propagation. With this setup, the feature maps of all preceding layers are combined and used as input to the current layer. Since the accumulated feature size gets bigger after each layer and demands a high computational effort, we decide to compress the concatenated features by a feature bottle-neck layer (FBN). FBN consisting of a $1{\times}1{\times}1{\times}C$ convolutional layer followed by a PReLU activation layer will reduce the dense-feature size to C channels before inputting to a channel attention-based residual layer (CAR) as in Fig. 5.3.

For the local residual learning block, we follow RCAN [120] to integrate channel attention [121] for adaptively scaling residual features. As shown in [120], this technique improves the reconstruction quality of high-resolution images. The architecture of *CAR* is comprised of a well-known feature extraction combination *conv-prelu-conv* followed by a channel attention weighting block (CAW). CAW starts with a global average pooling layer which collapses an input feature from $W{\times}A{\times}H{\times}C$ to $1{\times}1{\times}1{\times}C$. It is then reshaped to $1{\times}C$ and is followed by a 1D down-sampling convolution ($1{\times}C_r$) and 1D up-sampling convolution ($1{\times}C$). Here $C_r = \frac{C}{r}$ with r is a predefined reduction ratio. After going through a sigmoid activation layer, the $1{\times}C$ scaling weight is reshaped and broadcasted to the form $W{\times}A{\times}H{\times}C$ being ready for an element-wise multiplication with the input feature.

5.2.2.2 Attention-based Multi-Path Learning

This part is comprised of two separate learning paths targeting spatial and angular aspects of the feature maps. Each path includes an FBN, two SFEs, and an attention-based weighting block as in Fig. 5.3. There are two types of attention-based weighting, one is for spatial feature dimensions denoted as SAW and the other is for angular feature dimension denoted as AAW. FBN is employed to reduce the size of densely connected feature-maps generated by the residual learning part. After this block, the feature size is reduced from $(R + 1)C$ channels to C channels. The reduced feature map is fed to an SFE whose output is refined by an element-wise multiplication with attention weights computed by AAW or SAW. After the second SFE, the feature maps from the two paths are concatenated and squeezed to form the final residual feature map. The high-resolution EPI volume is acquired by adding up the residual information to the preliminarily up-sampled volume.

SAW consists of a global pooling, a 2D convolution, and a sigmoid activation layer. In global pooling layer, Average Pooling (P_{avg}) and Max Pooling (P_{max}) are applied to the input feature $F \in \mathbb{R}^{W{\times}A{\times}H{\times}C}$ and results in $F_{avg},\ F_{max} \in \mathbb{R}^{W{\times}1{\times}H{\times}1}$. These feature maps are then concatenated and reshaped to form the global pooling feature $F_{pool} \in \mathbb{R}^{W{\times}H{\times}2}$. We then applied a 2D convolution with kernel size 5×5×1 followed by a sigmoid activation function. These weighting values are then reshaped and broadcasted to the form $W{\times}A{\times}H{\times}C$ for an element-wise multiplication. The following equation summarizes the computation of spatial attention weights.

$$W_s(F) = \sigma\Big(f_{5\times5\times1}\Big(\big[P_{avg}(F); P_{max}(F)\big]\Big)\Big) \tag{5.2}$$

AAW consists of a global pooling, a fully connected, and a sigmoid activation layer. Given an input feature $F \in \mathbb{R}^{W{\times}H{\times}A{\times}C}$, we follow a similar procedure as of SAW to compute global pooling features $F_{pool} \in \mathbb{R}^{2A}$. Here, P_{avg} and P_{max} are applied to spatial and channel dimensions of the input feature map and the output features are concatenated along the angular dimension. We then compute A weighting values by applying a fully connected layer followed by a sigmoid activation function. Equation 5.3 outlines this computation.

$$W_a(F) = \sigma\Big(D_{2A:A}\Big(\big[P_{avg}(F); P_{max}(F)\big]\Big)\Big) \tag{5.3}$$

5.2.3 Preliminary Spatial Super-Resolution

Given an EPI volume $\boldsymbol{V}_l$ with a resolution of $W{\times}A{\times}H$ as an input, the output of PSSR is an EPI volume $\boldsymbol{V}_h$ with a resolution of $\zeta_z W{\times}A{\times}\zeta_z H$, where ζ_z is a spatial scaling factor (i.e., $2, 4$). The procedure of PSSR is as follows. First, 2D images (SAIs) along angular dimension ($\boldsymbol{V}_l(x, i, y), i = 1, 2, .., A$) are extracted from the input volume. Secondly, each image is separately super-resolved to the desired resolution by a SISR method. Finally, we combine these up-sampled images to form the output volume $\boldsymbol{V}_h$, see Fig. 5.2.

In this work, deep learning-based methods (i.e., EDSR [58], RCAN [120]) are employed to up-sample SAIs. With many advanced learning architectures introduced lately and sufficient training data, deep learning-based approaches easily outperform optimization-based approaches and provide state-of-the-art performance in the SISR task. However, as pointed out in the literature [57, 59–61] that SISR alone did not perform well on light field images due to missing the contribution of external information shared across multiple SAIs. This such information is well contained in an EPI volume and is exploited in the proposed EVRN to enhance the output of SISR. As will be shown later in the experimental result, the proposed approach substantially improves the reconstruction quality of SISR on both challenging synthetic and real-world LF data.

5.2.4 Preliminary Angular Super-Resolution

In this stage, a novel perspective image is generated for each consecutive pair of SAIs in the input volume. This task is done by a novel view synthesis module (NVS) as depicted in Figure 5.2. Given an EPI volume $\boldsymbol{V}_l$ with a resolution of $W{\times}A{\times}H$ as an input, the output of PASR is an EPI volume $\boldsymbol{V}_h$ with a resolution of $W{\times}(2A-1){\times}H$. The angular resolution of the volume is up-sampled by a scaling factor of $\zeta_\theta = \frac{2A-1}{A}$.

The synthesis of a novel view can be seen as the interpolation of a novel pixel along a line on an EPI image, see Fig. 5.1. This task is indeed trivial if the slope of this line, also referred to as disparity value [24], is known in advance. Therefore, many previous approaches [24, 48, 53] proposed to firstly estimate disparity maps and then exploit them for synthesizing novel views. However, as shown in [22, 52], this explicit estimation of a disparity map is not necessary since a learning-based approach which directly inferences novel views can already provide better performance. In this work, two different approaches of NVS are evaluated, i.e., nvs-cnn and nvs-mean. nvs-cnn is the proposed end-to-end CNN learning to synthesize a novel view from an input image pair. In nvs-mean, the new view is computed by simply averaging the two input images. As will be shown in Section 5.3.4,

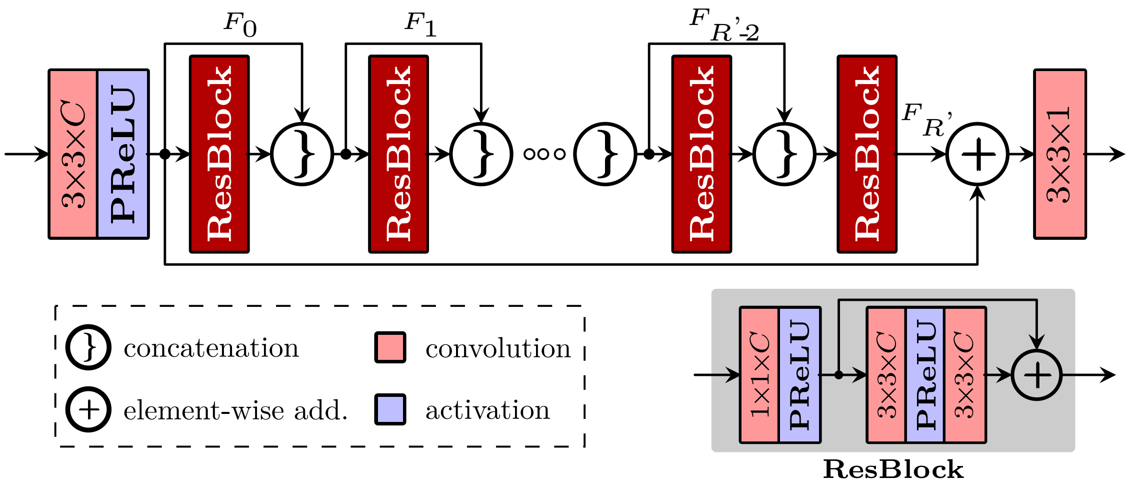


Figure 5.4: The network architecture of the proposed CNN-based method for PASR

this straightforward approach can provide good results in the case of narrow baseline LF images captured by plenoptic cameras [11, 12].

The network architecture of nvs-cnn is presented in Fig. 5.4. Similar to EVRN, global/local residual learning and dense connection are employed in the proposed network. We stack two input images to form an input feature $X \in \mathbb{R}^{W \times H \times 2}$. A shallow feature extraction layer consisting of a 2D convolutional kernel $3 \times 3 \times C'$ followed by a PReLU activation is applied to X. This results in a feature map $F_0 \in \mathbb{R}^{W \times H \times C'}$. After going through R' layers of residual blocks, we get a residual feature map $F_{R'} \in \mathbb{R}^{W \times H \times C'}$ which is added to F_0 and then squeezed by a 2D convolution kernel $3 \times 3 \times 1$ to acquire a novel perspective image ($I \in \mathbb{R}^{W \times H}$). For residual block, we use a simple design that consists of a bottleneck layer, i.e., 2D convolution kernel ($1 \times 1 \times C'$) followed by a PReLU, and a common combination conv-prelu-conv as in Fig. 5.4.

5.3 Experimental Results

5.3.1 Dataset and Training

Seven published light field datasets, listed in Table 5.1, are employed for training and testing the proposed approach. There are three synthetic datasets generated by 3D object models and blender software [4, 14, 15]. The other four datasets are real-world data captured by Illum cameras [16–18] and a gantry setup [19]. While synthetic scenes have

Table 5.1: Summary of training and test dataset

Datasets	Type	Angular	Training	Test
HCI13[14]	synthetic	9×9	5	2
HCI17[4]	synthetic	9×9	20	4
InSyn[15]	synthetic	9×9	31	8
StGantry[19]	real-world	17×17	9	3
StLytro[16]	real-world	14×14	250	53
InLytro[17]	real-world	15×15	28	8
EPFL[18]	real-world	15×15	81	12
Sum	-	-	424	90

the same angular resolution of 9×9, the angular resolution of real-world scenes varies from 14×14 to 17×17. To have an uniform light field data for training and testing, we follow the previous work [57, 59–61] to remove the border views and keep only 9×9 sub-aperture images in the middle.

5.3.1.1 Training EVRN

The LF images in the training set are transformed into YCbCr color space and only Y channel data is used for training. In the inference phase, we apply the trained network to each color space separately and then convert them back to RGB color image. Each LF image is spatially cropped into 4D patch ($P_H \in \mathbb{R}^{48\times48\times9\times9}$) with a stride of 16 pixels. Plain patches which do not include texture information are ignored. For each 2D patch with the size of 48×48 from P_H, we apply bicubic down-sampling with a scaling factor $\zeta_{\mathbf{z}}$ ($\zeta_{\mathbf{z}} = 2, 4$) to acquire a low spatial resolution patch $P_{LS} \in \mathbb{R}^{\zeta_{\mathbf{z}}^{-1}48\times\zeta_{\mathbf{z}}^{-1}48\times9\times9}$. Each 2D patch in P_{LS} is then up-sampled using SISR method to generate a spatial pre-scaling 4D patch P_S. A low angular resolution 4D patch ($P_{LA} \in \mathbb{R}^{48\times48\times5\times5}$) is extracted from P_H by removing 2D patches at angular position (ρ_i, τ_i), where $(\rho_i \bmod 2) \wedge (\tau_i \bmod 2) = 0$. We then up-sampled P_{LA} using the technique discussed in Section 5.2.4 to acquire angular pre-scaling 4D patch P_A. An angular-spatial pre-scaling 4D patch P_{SA} is generated by applying the same procedure to P_S. At this point, we have three 4D patch pairs $(\{P_S\}, \{P_H\})$, $(\{P_A\}, \{P_H\})$, and $(\{P_{SA}\}, \{P_H\})$ to train EVRN for SSR, ASR, and ASSR respectively. For each 4D patch pair, e.g., $(\{P_S\}, \{P_H\})$, a 3D EPI volume pair $(\{V_L\}, \{V_H\})$ is formed by extracting and combining the horizontal and vertical EPI volumes from each 4D patch set.

Padding is enabled in all 3D convolution layers to reserve the resolution. The number of

Table 5.2: Performance analysis of various model configurations.

Models	Synthetic		Real-world	
	PSNR	SSIM	PSNR	SSIM
model 1	38.61	0.9613	38.09	0.9581
model 2	38.63	0.9613	38.13	0.9588
model 3	38.77	0.9627	38.21	0.9592

Table 5.3: Ablation investiation of attention modules (CAS, AAS, SAW). The average PSNRs are computed for dataset EPFL with scaling factor ×2 after 200 epochs.

Modules	Combination of attention modules							
CAW	✗	✓	✗	✓	✗	✓	✗	✓
SAW	✗	✗	✓	✓	✗	✗	✓	✓
AAW	✗	✗	✗	✗	✓	✓	✓	✓
PSNR	36.92	36.95	36.93	36.95	36.94	36.97	36.94	36.98

residual blocks R and channel size C were set to 7 and 64 respectively. ℓ_1 loss function was used since it provides better results for SR in the proposed approach. The proposed network was implemented in TensorFlow running on a PC with an Nvidia 1080Ti GPU. As an optimizer, a variation of the Adam optimizer, AdamW [122], was used. AdamW adds a weight decay as regularization to the Adam optimizer. The learning rate was initialized to 2×10^{-4} and decreased by a factor of 2 after 10 epochs. The *weight decay* was set to 10^{-4}. For the initialization of convolution parameters, *Glorot uniform initializer* was used. The bias of the CNN layers and the PReLU parameters were initialized to zero.

5.3.1.2 Training NVS-CNN

To prepare the training data for NVS-CNN, we first extracted EPI volumes from 4D patches P_H. For each EPI volume, we use even-index views as ground-truths and the two neighbor views as inputs. This gives us 4 training pairs for each EPI volume. We empirically set the number of residual blocks R' and the number of feature channels C' to 7 and 64 respectively. Padding is enabled in all convolution layers to preserve the spatial dimension. For this training, we employ Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$). The learning rate is set to 2×10^{-4} which is halved after every 10 epochs.

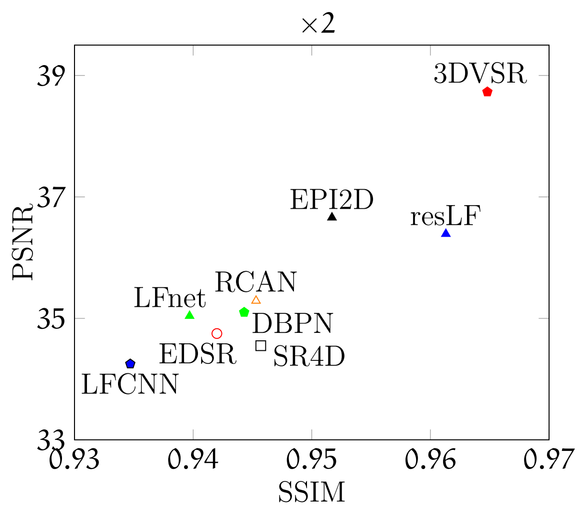
5.3.2 Model Analysis

To analyze the contribution of attention modules to the performance of the proposed approach, we conducted an experiment in which three models were tested. In the first model, all attention modules were removed from the network. The second model includes only the channel attention module (CAW), and the third model includes all attention modules. These models were trained for SSR tasks ($\zeta = 2$) and their results are listed in Table 5.2. It can be seen from the table that attention modules help to improve the reconstruction quality. Without attention modules, *model 1* scores 38.61 dB and 38.09 dB on average for synthetic and real-world light field data respectively. These figures slightly increase in *model 2* where the CAW module is included. Employing attention modules in both multi-path learning (AAW, SAW) and residual learning (CAW) provides the highest quality in terms of PSNR and SSIM.

A more comprehensive ablation study of attention modules can be found in Table 5.3. In this experiment, we investigated the effects of various combinations of attention modules. The eight networks were trained for spatial super-resolution application with scaling factor $\times 2$ and have the same configuration of residual blocks (R=4) and channel size (C=16). RCAN was employed for the PSSR stage, and the angular size of EPI volume was set to 3. After 200 epochs, we evaluate the performance of trained networks on the EPFL dataset and report the average PSNR values. From Table 5.3, it can be seen that the baseline network (without any attention modules) gives the lowest PSNR value (36.92dB). Furthermore, we observed that channel attention (CAW) and angular attention (AAW) demonstrate a clear contribution to the performance of EVRN. While SAW itself provides not much improvement, its combination with AAW and CAW delivers the best performance (36.98dB).

5.3.3 Spatial Super-Resolution

In this section, the evaluation results of 3DVSR applied to the SSR problem are discussed. We employed EDSR [58] and RCAN [120] for preliminary spatial super-resolution and tested against two scaling factors $\zeta_z = 2$ and $\zeta_z = 4$. Nine state-of-the-art approaches are selected for quantitative and qualitative comparisons. Among them, there are three SISR approaches (EDSR [58], DBPN [123], RCAN [120]) and six approaches provided for 4D LF (pcabm [54], LFnet [59], LFCNN [52], EPI2D [57], SR4D [60], resLF [61]). The result of bicubic interpolation is presented as a baseline result. All methods, except LFCNN, EPI2D, and LFnet, were tested using their released codes and pre-trained models. For EPI2D and LFnet, since the authors did not publish their source codes, we followed their papers to



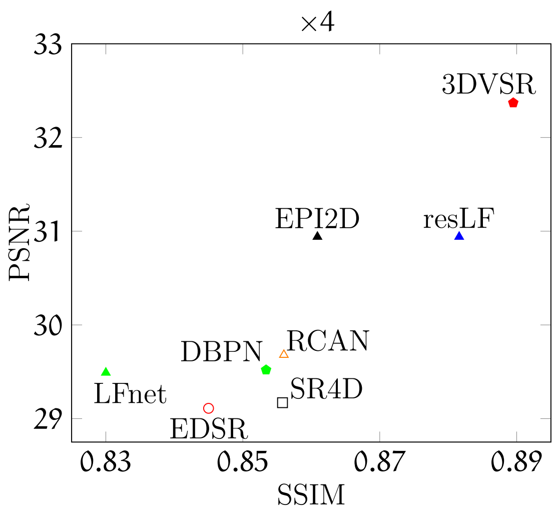


Figure 5.5: Spatially super-resolution results achieved by state-of-the-art SSR methods on 7 public datasets. The proposed approach outperforms single image SR methods (EDSR [58], RCAN [120], DBPN [123]) and LF SSR methods (pcabm [54], LFCNN [52], LFnet [59], EPI2D [57], SR4D [60], resLF [61]).

implement the models. Although the source code of LFCNN is available, its pre-trained model is not provided. Therefore, we retrained LFCNN, as well as EPI2D and LFnet, using our training dataset.

Table 5.4 and Figure 5.5 list quantitative results for ×2 and ×4 in terms of PSNR and SSIM metrics. For each scene, the quality metrics are calculated as an average of 7×7 SAIs in the middle. These values are then averaged over the test dataset. The two configurations of PSSR using EDSR and RCAN are denoted as 3DVSR-EDSR and 3DVSR-RCAN respectively. It can be seen from the table that the proposed approach scores the best PSNR and SSIM value for both ×2 and ×4 problems on average. Compared to SISR approaches EDSR and RCAN, 3DVSR respectively improves the reconstruction quality by 3.87dB and 3.63dB for ×2 and 3.13dB and 2.67dB for ×4. This improvement pays a tribute to the proposed enhancement network (EVRN) which exploits EPI volume structure to correct the high-frequency information from the output of SISR. The advantage of using EPI volume is also evident when compared to the 2D EPI-based approach [57]. Using a similar SISR technique (i.e. EDSR), the proposed approach scores 1.96 dB and 1.4dB better on average for ×2 and ×4 respectively.

Qualitative comparisons of six light field SSR approaches are shown in Fig. 5.6. It can be observed from the figures that the proposed approach shows superior performance in visual effects for both synthetic and real-world scenes. For example, only 3DVSR can

Table 5.4: Quantitative comparison of SSR approaches in terms of reconstruction quality measured by PSNR/SSIM. The results of two up-scaling factor (×2, ×4) on seven public datasets are reported.

Approach	Scale	HCI17[4]	HCI13[14]	InLytro[17]	InSyn[15]	EPFL[18]	StGantry[19]	StLytro[16]	avg
Bicubic	×2	28.33/0.876	35.02/0.951	30.64/0.900	29.18/0.892	28.30/0.850	33.52/0.965	31.83/0.924	30.97/0.908
pcabm[54]	×2	28.94/0.886	35.56/0.949	31.10/0.901	29.69/0.894	28.80/0.853	30.10/0.885	32.05/0.920	30.89/0.898
EDSR[58]	×2	31.51/0.926	39.57/0.973	33.27/0.932	33.70/0.932	30.50/0.890	39.06/0.985	35.62/0.957	34.75/0.942
DBPN[123]	×2	31.93/0.930	39.90/0.974	33.50/0.934	34.27/0.935	30.80/0.893	39.30/0.985	36.00/0.959	35.10/0.944
RCAN[120]	×2	32.25/0.932	39.91/0.975	33.57/0.934	34.39/0.936	30.90/0.894	39.82/0.986	36.21/0.960	35.29/0.945
LFCNN[52]	×2	31.67/0.904	38.03/0.963	33.48/0.928	32.12/0.906	32.22/0.926	37.30/0.974	34.91/0.941	34.25/0.935
LFnet[59]	×2	32.31/0.913	39.04/0.968	34.06/0.931	33.13/0.916	32.95/0.931	38.27/0.976	35.53/0.943	35.04/0.940
EPI2D[57]	×2	33.63/0.929	41.18/0.976	35.15/0.943	35.11/0.930	34.14/0.943	40.36/0.986	37.04/0.956	36.66/0.952
SR4D[60]	×2	32.49/0.943	39.99/0.977	33.96/0.944	32.29/0.929	30.21/0.889	35.36/0.965	37.53/**0.972**	34.55/0.946
resLF[61]	×2	35.14/**0.956**	39.84/0.974	35.43/**0.956**	35.27/0.948	34.36/0.954	37.04/0.975	37.67/0.966	36.39/0.961
3DVSR-EDSR	×2	**35.91**/0.954	**43.26**/**0.984**	**37.38**/0.955	**37.01**/**0.948**	**36.53**/**0.955**	**41.53**/**0.988**	**38.74**/0.965	**38.62**/**0.964**
3DVSR-RCAN	×2	**36.08**/**0.955**	**43.20**/**0.984**	**37.46**/**0.955**	**37.02**/**0.949**	**36.77**/**0.956**	**41.76**/**0.988**	**38.83**/**0.966**	**38.73**/**0.965**
Bicubic	×4	24.06/0.694	30.15/0.857	26.66/0.775	24.77/0.765	24.88/0.706	27.37/0.860	26.84/0.790	26.39/0.778
pcabm[54]	×4	24.65/0.726	30.77/0.865	27.14/0.792	25.37/0.780	25.37/0.725	26.51/0.790	27.32/0.804	26.73/0.783
EDSR[58]	×4	26.17/0.784	33.83/0.905	28.78/0.830	28.05/0.842	26.67/0.766	30.97/0.932	29.30/0.857	29.11/0.845
DBPN[123]	×4	26.58/0.797	34.19/0.912	29.05/0.835	28.83/0.854	27.01/0.776	31.34/0.936	29.68/0.864	29.52/0.853
RCAN[120]	×4	26.70/0.801	34.46/0.913	29.07/0.837	28.88/0.856	27.15/0.777	31.66/0.940	29.81/0.867	29.68/0.856
LFCNN[52]	×4	26.07/0.700	31.47/0.859	28.35/0.796	26.51/0.756	27.22/0.783	28.61/0.861	28.17/0.789	28.06/0.792
LFnet[59]	×4	27.16/0.745	33.33/0.890	29.41/0.829	28.02/0.799	28.43/0.818	30.52/0.900	29.56/0.829	29.49/0.830
EPI2D[57]	×4	28.15/0.786	35.60/0.914	30.56/0.851	29.66/0.838	29.64/0.847	32.30/0.935	30.70/0.856	30.94/0.861
SR4D[60]	×4	27.15/0.830	34.08/0.921	29.61/0.864	27.47/0.839	26.84/0.786	28.26/0.854	30.75/**0.896**	29.17/0.856
resLF[61]	×4	28.83/**0.844**	34.30/0.917	30.63/**0.881**	29.79/**0.875**	29.53/0.869	30.63/0.903	30.80/0.882	30.64/0.882
3DVSR-EDSR	×4	**29.59**/0.833	**37.07**/**0.938**	**31.63**/0.875	**31.13**/0.870	**30.85**/**0.872**	**33.36**/**0.946**	**32.07**/0.883	**32.24**/**0.888**
3DVSR-RCAN	×4	**29.70**/**0.836**	**37.17**/**0.939**	**31.69**/**0.876**	**31.23**/**0.872**	**31.09**/**0.872**	**33.53**/**0.948**	**32.15**/**0.885**	**32.37**/**0.890**

(red: best, blue: second best)

reconstruct a correct pattern of wooden sticks in *general_55* and clear detail of two faces in *Smilling_crowd*. In pcabm [54], the low-resolution light field image is divided into sets of patches that are then super-resolved separately. Although the overlapping of patches helps to smooth the output images, the inconsistencies in reconstruction quality between patches are unavoidable and lead to blocky artifacts (i.e. *ISO_Chart*, *Smilling_crowd*). In LFCNN [52], each SAI is super-resoluted independently using a SISR approach [51] whose CNN is quite trivial and shallow. LFCNN's results are, therefore, over-smoothed in all the test scenes. SR4D [60] and resLF [61] include multiple SAIs in their super-resolution process. This allows them to exploit external information to reconstruct high-resolution images. However, these approaches still fail to reconstruct high quality images in challenging light field scenes with noisy and high-frequency pattern (i.e. *general_55*) or highly degraded content (i.e., ×4 down-sampled: *ISO_Chart* and *Smilling_crowd*). From the figure, their results are ambiguous and with obvious artifacts. Compared to EPI2D, we follow a similar approach in which the output of SISR is enhanced by reinforcing EPI structures of light field images. However, instead of relying on a narrow and feature-limited EPI as in EPI2D, we proposed to refine an EPI volume that allows us to exploit global information across multiple EPIs. Our results, therefore, show significantly better image qualities where the

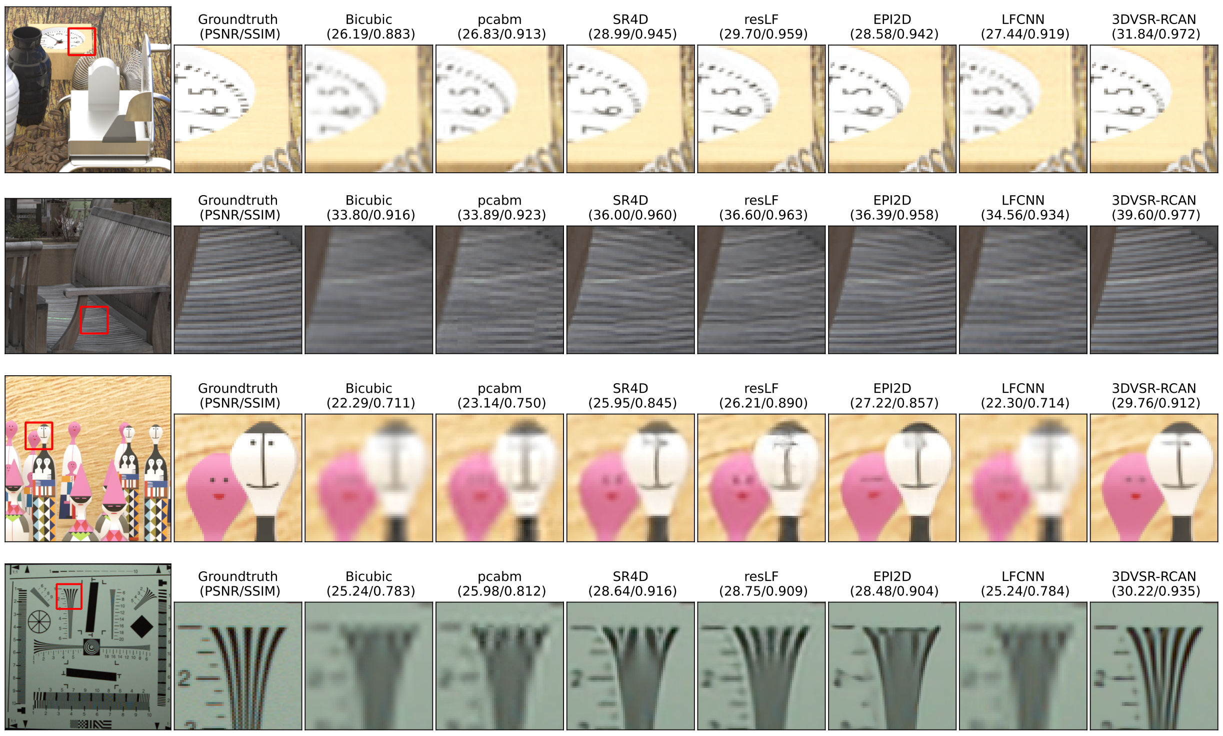


Figure 5.6: Qualitative comparison of SSR approaches on two scaling factors (×2,×4). The SAI at $\boldsymbol{\theta} = (4, 4)$ is visualized together with its zoom-in region marked by a red rectangle. The first two rows: ×2 results of synthetic scene *coffee_beans_vases* [15] and real-world scene *general_55* [16]. The last two rows: ×4 results of synthetic scene *smiling_crowd* [15] and real-world scene *ISO_Chart* [18].

texture is recovered correctly and with more high-frequency details.

To investigate the reconstruction quality concerning different perspectives, we conducted an experiment in which the scene *two_vases* is used as input to perform both ×2 and ×4 SSR. For each view, the PSNR and SSIM values are computed and the results are visualized in Fig. 5.7. Here, we compared our results with the results of the state-of-the-art approach ResLF [61]. resLF employs a star-like structure of SAIs as input to super-resolute a single SAI lying in the center. This strategy is indeed beneficial to reconstruct high-resolution images. As in Table 5.4, resLF surpasses EPI2D and SR4D for the most of test dataset. However, the disadvantage of resLF's approach is the unequal treatment of SAI at different perspectives. The SAI close to the border has fewer input images than the SAI close to the center. Therefore, reconstruction qualities of non-central views are relatively lower. In contrast, the proposed approach jointly uses EPI information and spatial information to reconstruct an EPI volume and thus achieves much higher output qualities with more stable distribution across all perspectives.

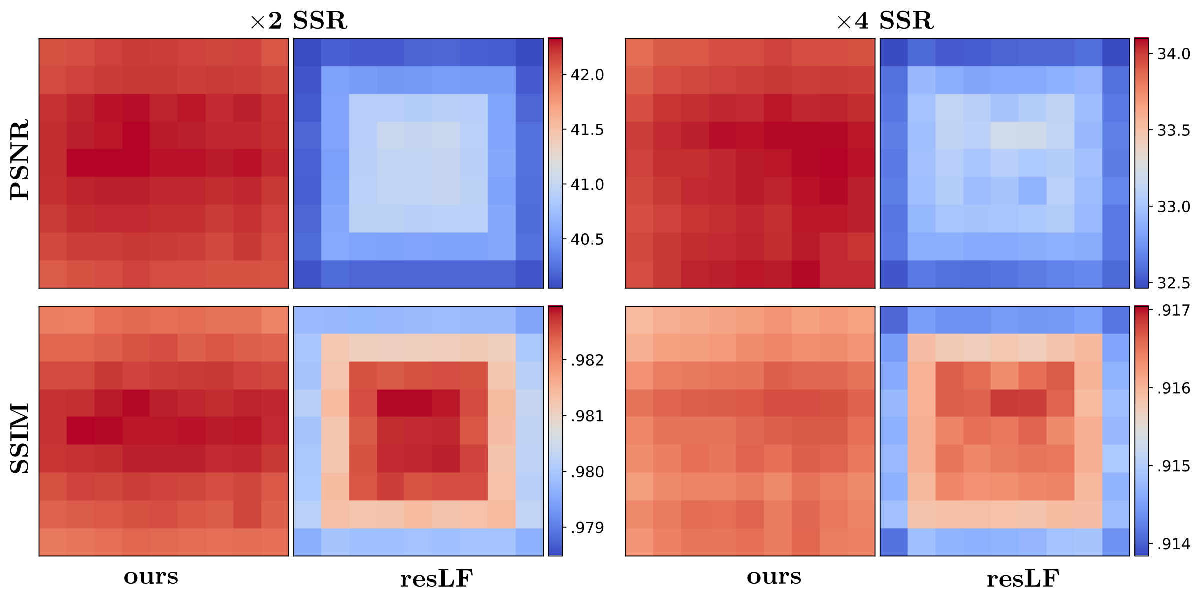


Figure 5.7: Visualization of PSNR and SSIM values on each SAI of light field scene *two_vases* [15]. Compared to ResLF [61], the proposed approach achieve a better reconstruction quality while maintaining a consistent performance on all perspective images.

Fig. 5.8 presents a quantitative and qualitative comparison of EPI volumes reconstructed after the first stage (PSSR) and the second stage (EVRN). To better justify the improvement in angular dimension, we extract EPI images for each output and compare them to the EPI of the ground truth. From the figure, it is evident that the EVRN substantially improves the quality of reconstructed volume in both spatial and angular dimensions.

5.3.4 Angular Super-Resolution

This section discusses the evaluation of the proposed approach for the angular super-resolution of light field images. The seven datasets, listed in Table 5.1, are employed in this evaluation. For each scene in the test set, the angular resolution is down-sampled from 9×9 to 5×5 while the spatial resolution remains intact. With these low-resolution LFs as inputs, we then reconstruct the original size LFs following the procedures discussed in Section 5.2.4. This means that 56 missed perspective images are reconstructed from 25 input images. In the PASR stage, we tested two approaches, one generates novel views by averaging and the other uses an end-to-end CNN. The results of these two PASR approaches as well as the final results after applying the refinement network are reported. We compare our approach with

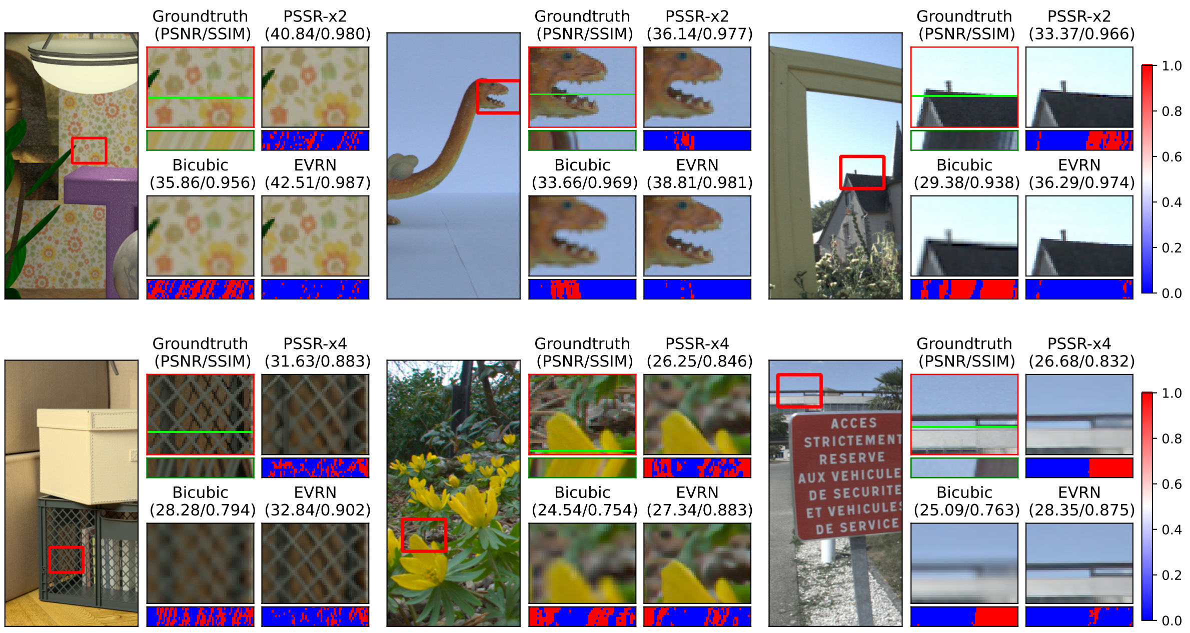


Figure 5.8: Comparison of two stages in reconstruction of spatially high resolution LF. The SAI at $\theta = (4, 4)$ is visualized together with its zoom-in region marked by a red rectangle. For each approach, an EPI at horizontal line marked in green is extracted and compared to the EPI of Groundtruth. The first row, from left to right: ×2 results of synthetic scene *mona* [14] and real-world scenes *Ankylosaurus_&_Diplodocus_1*, *Framed* [18]. The second row, from left to right: ×4 results of synthetic scene *boxes* [4] and real-world scenes *Flowers*, *Sign* [18].

three previous approaches (vsyn [53], LFCNN [52], and LFSR [22]).

Table 5.5 lists quantitative results of ASR approaches running on the 7 public datasets. We employed PSNR and SSIM as quality metrics that are computed for newly generated SAIs and are averaged over all scenes in each dataset. *nvs-mean* and *nvs-cnn* denotes the two PASR approaches. *nvs-mean* computes a novel perspective image by averaging two neighboring images, while *nvs-cnn* inferences a novel image using a residual CNN as depicted in Fig. 5.4. The outputs of *nvs-mean* and *nvs-cnn* enhanced by EVRN are denoted as *3DVSR-mean* and *3DVSR-cnn* respectively. LFSR [22] has a strict requirement of supported angular resolution. This approach employs a fully-connected network in its output layer that always produces an angular-resolution of 14×14. For this reason, only real-world datasets [16–19] are tested with this approach. From Table 5.5, it can be seen that the proposed approach provides the highest reconstruction quality. 3DVSR improves PSNR and SSIM values by a large margin as compared to the previous approaches (i.e. a minimum of 3dB improvement in all test datasets). In narrow baseline light field image captured by a plenoptic camera [16–18], the difference between two neighboring views

Table 5.5: Quantitative comparison of ASR approaches on 7 light field datasets

Approach	HCI17[4]	HCI13[14]	InLytro[17]	InSyn[15]	EPFL[18]	StGantry[19]	StLytro[16]	avg
nvs-mean	29.23/0.878	42.25/0.987	40.35/0.979	27.73/0.828	37.81/0.975	25.53/0.728	40.35/**0.981**	34.75/0.908
nvs-cnn	37.02/0.971	44.15/0.992	40.49/0.975	38.76/0.979	37.89/0.974	30.02/0.847	40.75/0.980	38.44/0.960
vsyn[53]	23.34/0.665	29.41/0.764	32.03/0.899	21.40/0.613	27.76/0.794	19.35/0.542	30.34/0.887	26.23/0.738
LFSR[22]	-/-	-/-	39.91/0.980	-/-	37.63/0.979	22.24/0.688	39.57/0.977	34.84/0.906
LFCNN[52]	30.97/0.883	38.78/0.967	35.36/0.948	29.09/0.834	34.43/0.949	26.01/0.739	36.41/0.957	33.01/0.897
Wang18[124]	31.66/0.893	41.74/0.984	37.31/0.967	29.36/0.831	36.18/0.969	27.04/0.741	38.86/0.971	34.59/0.908
Wu19[125]	29.93/0.917	30.76/0.832	33.73/0.922	28.67/0.871	29.48/0.843	20.80/0.631	32.45/0.937	29.40/0.850
Wu19a[126]	33.19/0.936	43.28/0.985	39.90/0.969	32.12/0.899	37.68/0.966	27.51/0.746	40.03/0.973	36.24/0.925
3DVSR-mean	**40.12**/**0.979**	**47.07**/**0.994**	**44.22**/**0.985**	**40.32**/**0.979**	**43.64**/**0.987**	**32.30**/**0.854**	**43.41**/**0.985**	**41.58**/**0.966**
3DVSR-cnn	**40.00**/**0.981**	**45.15**/**0.994**	**43.04**/**0.982**	**40.90**/**0.982**	**42.30**/**0.986**	**32.11**/**0.861**	**42.20**/0.980	**40.81**/**0.967**

(red: best, blue: second best)

is almost invisible due to their sub-pixel displacement values (e.g. less than 0.5 pixel). In this case, a straightforward approach such as nvs-mean can score very well and the benefit of employing CNN in the PASR stage is limited, i.e., nvs-cnn provides less than 0.4 dB improvement as compared to nvs-mean. However, for the other test datasets, nvs-cnn presents a clear improvement as compared to nvs-mean (i.e., 1.9dB to 7.8dB). By exploiting the EPI volume structure to refine the results of the PASR stage, 3DVSR achieves significant improvements over nvs-mean and nvs-cnn by an average of 6.8 dB and 2.4 dB respectively. It is interesting to see that the performance of 3DVSR-mean is comparable to 3DVSR-cnn with a slightly better PSNR value (i.e. 0.7 dB). This demonstrates the superior performance of EVRN in reconstructing novel perspective images.

Fig. 5.9 shows the visual comparisons of evaluated ASR approaches on both synthetic and real-world light field scenes. vsyn [53] consists of two CNNs, one predicts auxiliary disparity maps and the other synthesizes novel views. It assumes that the disparity values of input light fields fall within a range that is quantized by a fixed step size. Based on the quantized disparity values, a cost volume is computed and is used as an input to the first CNN. The output disparity maps from the first CNN are used to pre-compute novel views that are then refined by the second CNN. The quality of the novel views highly depends on the accuracy of estimated disparity maps which in turn depend on the cost volume and assumed disparity range. From the figure, it can be seen that vsync does not generalize well and performs badly on light field scenes with a large disparity range (i.e. lego, big_clock, sideboard). Although the visual quality of the plenoptic scene (i.e., flowers_plants_36), for which vsyn was trained, is much better, it still suffers from the over-smoothed effect. In LFSR [22], novel views are generated by the mean of super-resolution of APIs. Since no spatial information is considered in its process, the results of LFSR are noisy and with obvious artifacts. Both LFCNN and our cnn-based PASR approach (nvs-cnn) follow a similar strategy in which novel perspective images are synthesized by using their surrounding

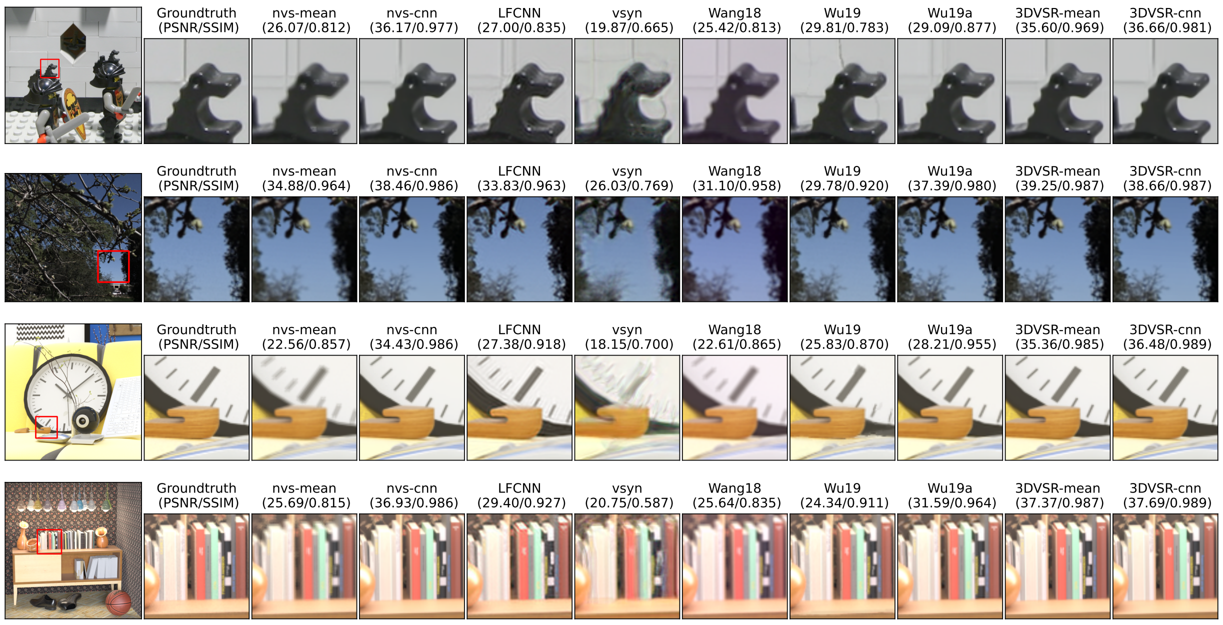


Figure 5.9: Qualitative comparison of ASR approaches on real-world and synthetic light field scenes. The SAI at $\boldsymbol{\theta} = (3, 3)$ is visualized together with its zoom-in region marked by a red rectangle. The first two rows: real-world scenes *lego* [19] and *flowers_plants_36* [16]. The last two rows: synthetic scenes *big_clock* [15] and *sideboard* [4].

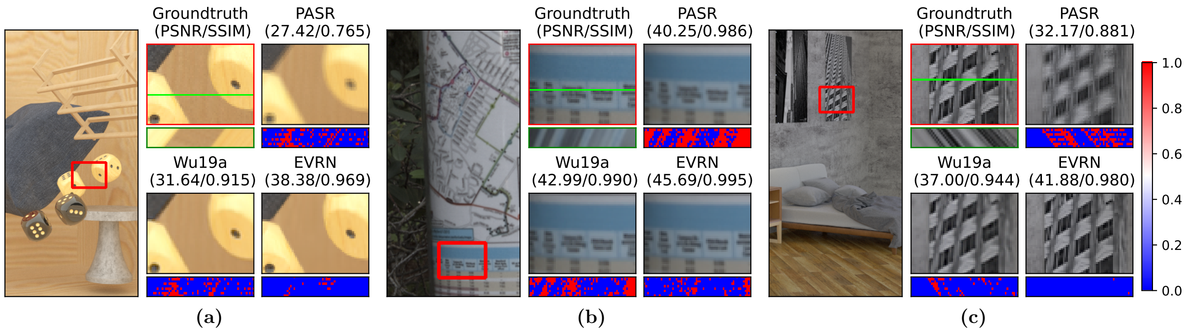


Figure 5.10: Comparison of two stages in reconstruction of angularly high resolution LF. The SAI at $\boldsymbol{\theta} = (3, 3)$ is visualized together with its zoom-in region marked by a red rectangle. For each approach, an EPI at horizontal line marked in green is extracted and compared to the EPI of Groundtruth. From left to right: synthetic scene *Flying_dice_dense* [15], real-word scene *general_29* [16], and synthetic scene *bedroom* [4].

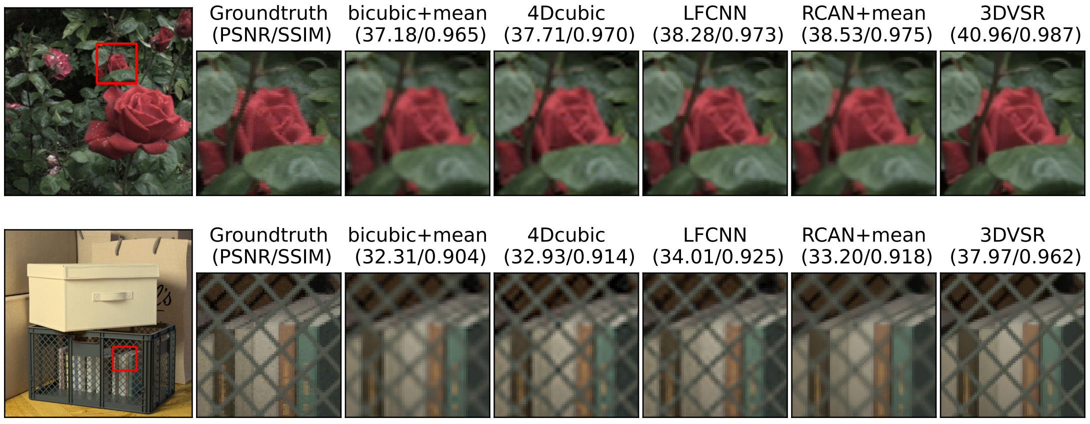


Figure 5.11: Angular-spatial super-resolution results of different approaches. The SAI at $\boldsymbol{\theta} = (3, 3)$ is visualized together with its zoom-in region marked by a red rectangle. top row: real-world scene *Rose* [17]; bottom row: synthetic scene *boxes* [14];

neighbor images. However, as opposed to the simple architecture of LFCNN which only consists of convolution layers and activation layers, nvs-cnn employed many effective deep-learning structures (i.e. global/local residual learning, dense connection). nvs-cnn, therefore, outperforms LFCNN with much better visual quality. Compared to nvs-cnn, the novel perspective images generated by nvs-mean are more ambiguous with over-smoothed regions and artifacts as the result of the averaging method. However, after being refined by EVRN the visual qualities of these images are significantly enhanced (i.e., 3DVSR-mean) and are comparable to the enhanced version of nvs-cnn (i.e., 3DVSR-cnn).

Fig. 5.10 compares the angular super-resolution results of Wu19a [126] and the proposed approaches after the first stage (PASR) and after the second stage (EVRN). In this experiment, nvs-mean is employed as a preliminary angular super-resolution approach, and the output volume of PASR is fed to EVRN for the final enhancement. Compared to our PASR, Wu19a provides a better reconstruction quality with sharper content and less angular error. However, the refined volume of EVRN is by far better than the output of Wu19a. As a result, we achieve a minimum of 3.3dB improvement in PSNR and less error in EPIs.

5.3.5 Angular-Spatial Super-Resolution

In the ASSR problem, the resolution of 4D light field images is super-resoluted angularly and spatially. In other words, it consists of a super-resolution of each given SAI and a synthesis of novel perspective images which have the same higher resolution. As discussed in Section 5.2.1, the proposed approach handle ASSR in two stages. The first stage consists of PSSR followed by PASR to generate a 4D light field with the desired resolution. This preliminary up-sampled light field is then enhanced by EVRN in the second stage to acquire the final output. To evaluate the proposed approach, we conducted an experiment in which the spatial scaling factor was set to $\zeta_z = 2$ and a similar ASR configuration as in Section V.D. was applied.

The spatial-angular super-resolution results are shown in Fig. 5.11. *bicubic+mean* denotes a baseline approach that employs bicubic interpolation for SSR and averages neighbor views to generate novel perspective images. *4Dcubic* denotes an approach in which cubic interpolation is applied on two spatial and two angular dimensions. *RCAN+mean* denotes the result of our preliminary stage which uses *RCAN* [120] for PSSR and *nvs-mean* for PASR. The result after an enhancement using EVRN is denoted as *3DVSR*. Compared to the baseline approach and 4Dcubic, LFCNN provides a small improvement in reconstruction quality. Although LFCNN achieves an increase of about 1dB, its improvement in visual quality is negligible. The results of LFCNN are still ambiguous and lack high-frequency information. A similar performance can be seen in the results of *RCAN+mean* which are mostly over-smoothed due to the effect of averaging views. Compared to these approaches, 3DVSR produces a significant improvement in PSNR value (i.e., a minimum of 2.4 dB and 3.9 dB in scenes *Rose* and *boxes* respectively) and an obvious enhancement in visual quality.

5.4 Summary

This chapter presents a deep learning-based light field super-resolution approach. Based on EPI volume structure, a 3D projected version of 4D light field, we proposed a 2-stage framework that effectively addresses various problems in light field super-resolution, i.e., ASR, SSR, and ASSR. While the earlier stage provides flexible options to up-sample the input volume to the desired resolution, the later stage consisting of an EPI volume-based enhancement CNN substantially improves the reconstruction quality of the high-resolution EPI volume. The proposed enhancement network built on 3D convolutional operations and efficient deep-learning structures, i.e., global/local residual learning, dense connection, multi-path learning, and attention-based scaling, effectively combines angular and spatial information from the 3D EPI volume structure to reconstruct high-frequency details. An extensive evaluation performed on 90 challenging synthetic and real-world light field scenes from 7 published datasets shows that the proposed approach outperforms state-of-the-art methods to a large extent for both spatial and angular super-resolution problems, i.e., an average PSNR improvement of more than 2.0 dB, 1.4 dB, and 3.14 dB in SSR×2, SSR×4, and ASR respectively. The reconstructed 4D light field demonstrates a balanced performance distribution across all perspective images and presents superior visual quality compared to the previous works.

Chapter 6

4D Light field Image Compression

This chapter presents a compression framework for light field images. The main idea is to exploit the similarity across sub-aperture images extracted from 4D light field data to improve encoding performance. For this purpose, we employed the variational optimisation approach to estimate the disparity maps from light field images (Chapter 2), which are then applied to a motion-compensated wavelet lifting scheme. The lifting scheme decomposes 4D light field images into high-/low-pass sub-band views in which the redundancy of image data is eliminated. Making use of JPEG2000 for coding all sub-band views as well as disparity maps, our approach can therefore support both lossless and lossy compression. GPU acceleration is proposed to the motion-compensated wavelet decomposition (MWD) task, which is the second most time-consuming computation after disparity estimation in the proposed coding framework. The coding framework is tested with both synthetic and real-world light field datasets. The experimental results show that our approach outperforms JPEG–LS and the direct application of JPEG2000 in both lossless and lossy compression scenarios. In addition, a significant gain in processing time is achieved by our GPU-accelerated implementation as compared to CPU-based implementation. Parts of the results of this chapter have been published in [127].

6.1 Motion-compensation Light Field Compression

Compared to traditional photography, light field photography presents the capability to capture both spatial and directional information of the scene. Since the development of commercial hand-held plenoptic cameras, e.g., Lytro or Raytrix [128], LF imaging has

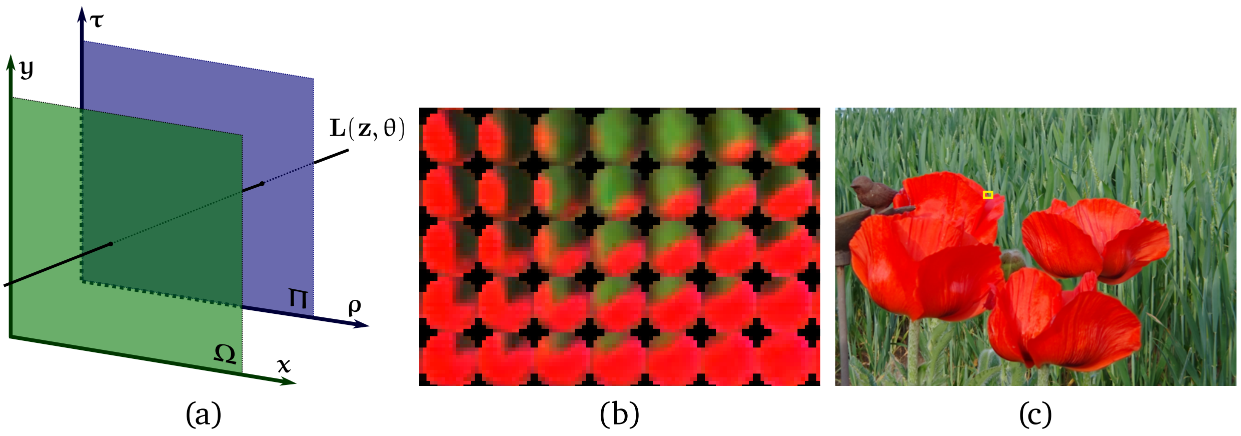


Figure 6.1: Lightfield representation and acquisition. (a) Two-plane parameterisation. (b) Lenslet-based sensor image. (c) Central sub-aperture image

been brought to a broader audience spanning from experts to casual users. The growth in popularity of LF imaging gives rise to the need to store, access and transfer LF data in a more efficient way. Due to its multi-dimension and rich-content property, LF images generally require more storage capacity, bandwidth and transferring time. For example, a single captured plenoptic-based LF image from a plenoptic-based camera, i.e., Raytrix R42 ([128]), comprises of 42Mpx 2D image which requires 252 Mbyte storage. Such an image when transferred in an uncompressed format results in a frame rate of 0.5 fps on a 1 Gbps network connection. The plenoptic image has a special structure distinguishing from the traditional 2D image, see Figure 6.1(b). It assembles a 2D array of micro-lens images which can be structured in the form of 4D array, Figure 6.1(a), and then re-arranged in the form of 2D array of sub-aperture images (SAIs), Figure 6.1(c). It is evident that there is a large similarity of scenery in sub-aperture images. Due to the small base-line property, the disparity value in plenoptic-based 4D LF is quite small (i.e., below 2 pixels). Therefore there is only a few pixel-shift from one SAI to the next neighbor SAI. This redundancy in LF image, specially plenoptic image, motivates the proposed encoding strategy.

Our coding framework is presented in Figure 6.2. Instead of directly coding raw light field images in the pixel domain, LF images are transformed into the wavelet domain before undergoing entropy coding. Specifically, the light field image is firstly decoded into multiple sub-aperture images, which is done by the *View Extraction* module, and a disparity map is then extracted from them by *Disparity Estimation* module. The light field toolbox developed by Dansereau et al. [71] is used for the first task and a variational optimisation approach proposed in Chapter 2 is applied for the second. The estimated disparity maps and all sub-aperture images are sent to the *Motion-compensated Wavelet Decomposition* (MWD) module to extract high-pass and low-pass sub-band views. These sub-

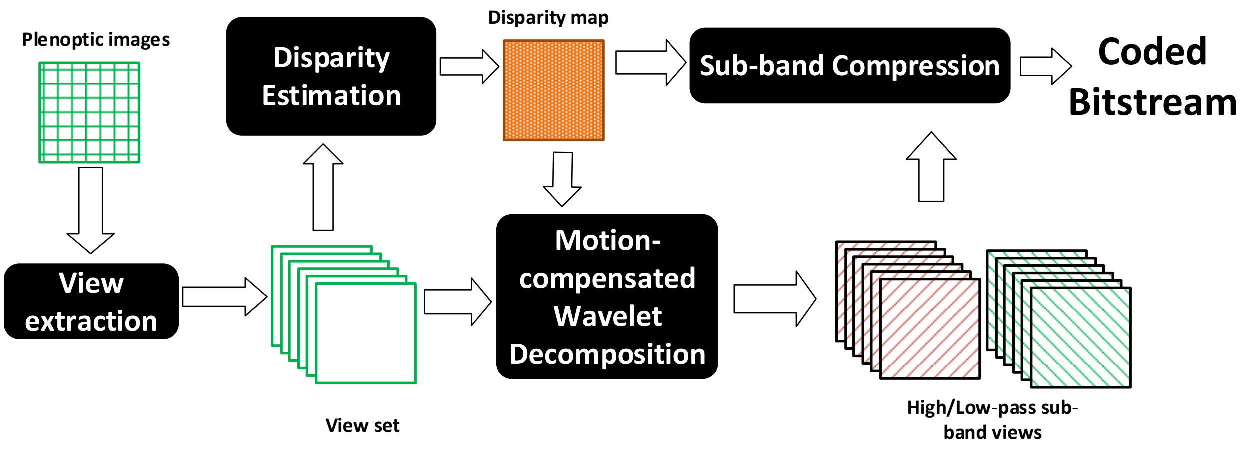


Figure 6.2: Lightfield coding framework

band views and the disparity maps are then encoded by JPEG2000 in *Sub-band Compression* module.

In our framework, the 2D wavelet transform and the motion compensation are jointly performed in order to effectively exploit the inter-view redundancy of sub-aperture images. The decomposition scheme is based on the lifting implementation of Discrete Wavelet Transform (DWT) [130]. In order to flexibly handle both lossless and lossy compression scenarios, we deploy a two-step lifting scheme based on Haar and bi-orthogonal 5/3 wavelet kernel. These two schemes are invertible. Therefore it allows us to reconstruct the original LF data from lossless bit-stream without losses. In the case that higher compression ratio is preferred, quantization can be applied in *Sub-band Compression* module before the entropy coding to further reduce the size of encoded data.

Consider a Light field $\boldsymbol{L}$ as a 2D grid of sub-aperture images defined on directional domain Π. Let $P \subset \Pi$ is a set of view indexes belonging to a row or a column of this grid. We apply the lifting scheme for each view $V_k(\mathbf{z}) = L(\mathbf{z}, \boldsymbol{\theta}_k), \boldsymbol{\theta}_k \in P$. Here $k = 0, 1, 2, \ldots$ is indexed from left to right or from top to bottom. A general form of a two-step lifting scheme is formulated in Equation 6.1,

$$\begin{aligned} h_k(\mathbf{z}) &= V_{2k+1}(\mathbf{z}) + \alpha_1 V_{2k}(\mathbf{z}) + \alpha_2 V_{2k+2}(\mathbf{z}) \\ l_k(\mathbf{z}) &= V_{2k}(\mathbf{z}) + \beta_1 h_k(\mathbf{z}) + \beta_2 h_{k-1}(\mathbf{z}) \end{aligned} \tag{6.1}$$

where $h_k(\mathbf{z})$ and $l_k(\mathbf{z})$ are high-pass and low-pass sub-band view, respectively. The weight parameters $(\alpha_1, \alpha_2, \beta_1, \beta_2)$ depends on wavelet kernel. For Haar and 5/3 transform, they are set to $(-1, 0, \frac{1}{2}, 0)$ and $(-\frac{1}{2}, -\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ respectively.

Let $\mathcal{W}_{i,j}[f]$ denote the motion-compensated mapping of 2D function $f(\mathbf{z})$ from the coordinate of view V_i to the coordinate of view V_j. The disparity-compensated version of this

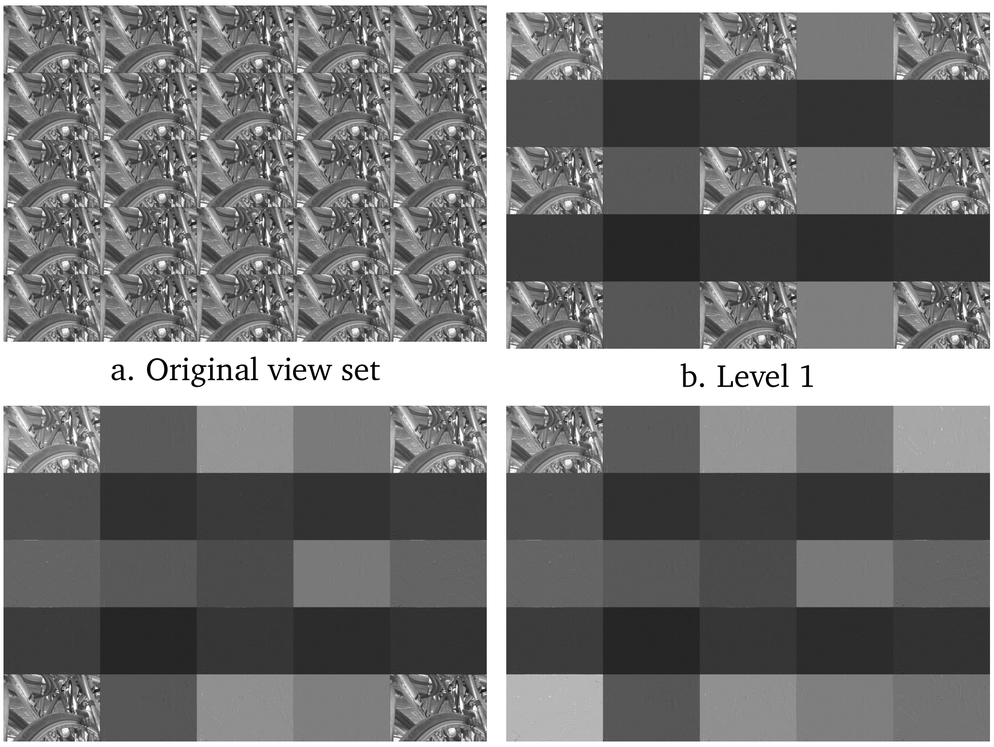


Figure 6.3: Demonstration of 3-level 2D MWD for a 5×5 view set of “bikes” lenslet data [129].

two-step lifting scheme is as follows

$$
\begin{aligned}
h_k(\mathbf{z}) =& V_{2k+1}(\mathbf{z}) + \alpha_1 \mathcal{W}_{2k,2k+1}[V_{2k}](\mathbf{z}) + \alpha_2 \mathcal{W}_{2k+2,2k+1}[V_{2k+2}](\mathbf{z}) \\
l_k(\mathbf{z}) =& V_{2k}(\mathbf{z}) + \beta_1 \mathcal{W}_{2k+1,2k}[h_k](\mathbf{z}) + \beta_2 \mathcal{W}_{2k-1,2k}[h_{k-1}](\mathbf{z})
\end{aligned}
\tag{6.2}
$$

SAIs with the index $2k$ and the index $2k+2$ are warped into the coordinate of the SAI at the index $2k+1$. Since the texture in three images are aligned, the weighted sum to compute high-pass sub-band view h_k results in an 2D array which are sparser than the one computed from Eq. 6.1. The inverse transform could be simply derived from Equation 6.2.

$$
\begin{aligned}
V_{2k+1}(\mathbf{z}) =& h_k(\mathbf{z}) - \alpha_1 \mathcal{W}_{2k,2k+1}[V_{2k}](\mathbf{z}) - \alpha_2 \mathcal{W}_{2k+2,2k+1}[V_{2k+2}](\mathbf{z}) \\
V_{2k}(\mathbf{z}) =& l_k(\mathbf{z}) - \beta_1 \mathcal{W}_{2k+1,2k}[h_k](\mathbf{z}) - \beta_2 \mathcal{W}_{2k-1,2k}[h_{k-1}](\mathbf{z})
\end{aligned}
\tag{6.3}
$$

The above 1D transform is extended to 2D transform by applying first the 1D transform horizontally and then vertically. In the case that there is more than one MWD level, the decomposition in the next level is computed on low-pass sub-band views from the current

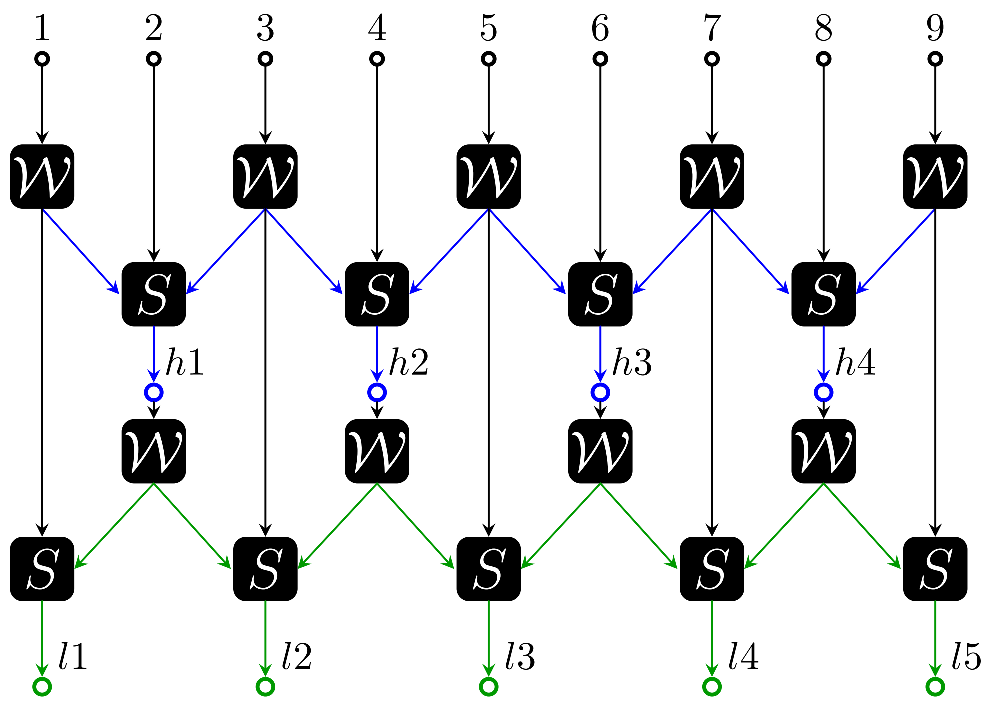


Figure 6.4: GPU-based computation of 1D MWD

level. Figure 6.3 illustrates three-level MWD of a light field image with 5×5 directional resolution.

6.2 Acceleration of Coding Framework

In the proposed coding framework, the most time consuming task is disparity estimation, in which disparity maps are extracted from 4D LF data by solving the variational optimization problem. The numerical computation of the iterative solvers and its GPU implementation are described in Chapter 2 and Chapter 3 respectively. The encoding of high/low-pass sub-band views in *sub-band compression* step is carried out by standard compression tools, i.e., JPEG2000, whose GPU-acceleration have been well addressed in the literature [131–133]. In this section, we are going to discuss the GPU implementation of MWD which comprises of warping operation and lifting scheme. The warping operation, as discussed in Chapter 3, is well-suited for GPU acceleration. The lifting scheme is in principle the weighted sums of 2D arrays and can be effectively implemented in the form of an OpenCL kernel.

Figure 6.4 depicts the GPU-accelerated computation of 1D MWD. The two OpenCL kernels are illustrated in a black background rectangle. $\mathcal{W}$ denotes the warping kernel and S

denotes the weighted sum kernel. Different from the warping kernel described in Chapter 3, which warps SAIs from all perspectives into the coordinate of the reference SAI, the warping kernel in MWD maps the input SAI into the coordinates of two neighbor SAIs. By combining two warping operations into a single kernel, we take advantage of texture cache to reduce retrieving time of pixel values from the input SAI stored in global memory. Kernel S takes three memory objects stored in global memory as inputs and performs a pixel-wise weighted sum of them. The global work-group sizes of the two kernels are then set to the horizontal and vertical sizes of SAI.

6.3 Experimental Results

In this section, we compare the compression performance of our proposed coding framework to JPEG–LS and JPEG2000 codecs. Both synthetic and real-world light field images are considered in our experiment. Five synthetic data are taken from the synthetic 4D light field dataset [14] and the other five real-world images are taken from JPEG pleno database [129]. The synthetic light field images have the directional resolution of 9×9 and the spatial resolution of 768×768 (except *horses* with 1152×512 resolution). All the real-world

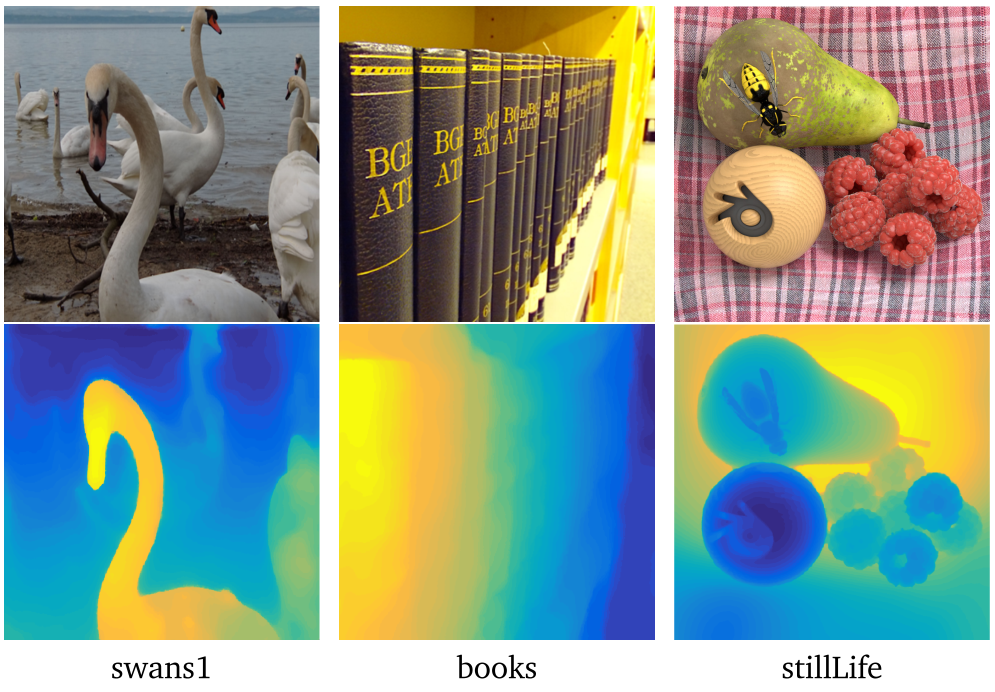
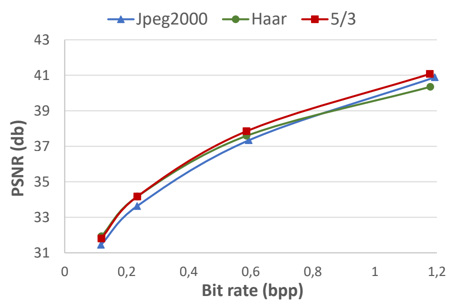


Figure 6.5: Estimated disparity map from light field data. *top* center view images. *bottom* color coded disparity map. Two images from the left are real-world data. The other is synthetic data.

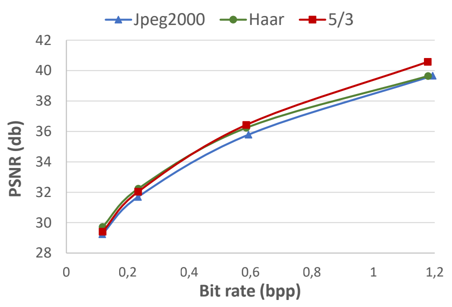
Table 6.1: Lossless compressed filesize in mega bytes (MB)

	Image	Raw	JpegLS	Jpeg2000	Haar	5/3
synthetic	buddha2	143.33	90.38	62.42	58.04	57.18
	mona	143.33	77.44	52.16	40.99	40.82
	papillon	143.33	70.17	51.57	44.47	44
	stillLife	143.33	113.86	79.59	67.79	66.5
	horses	143.33	96.41	60.8	59.93	58.65
real-world	books	183.09	93.84	103.74	89.18	86.19
	bikes	183.09	106.73	92.14	84.72	78.73
	danger	183.09	105.42	90.41	80.83	74.68
	pillars	183.09	91.48	76.87	71.21	66.74
	swans1	183.09	95.1	88.45	80.39	75.02

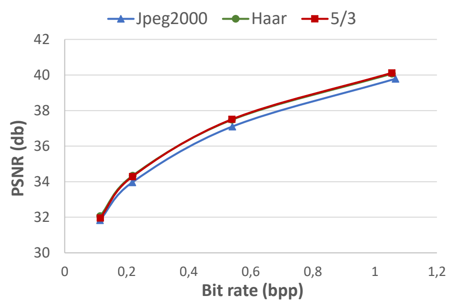
light field data are captured using Lytro Illum camera that provides 15×15 sub-aperture images and each with 434×625 spatial resolution. Because of the vignetting effect and the geometrical distortion of the microlens, the border views are either dark or contain less intensity similarity to the other views. Therefore MWD is applied only for the middle 13×13 views. The remaining views are directly encoded by *Sub-band Compression* module.



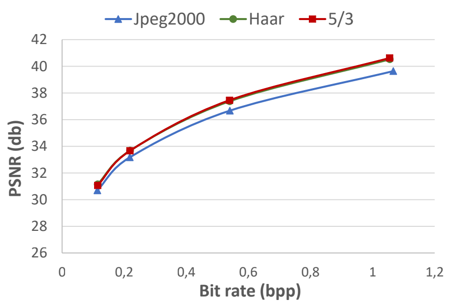
(a) $PSNR_{YUV}$



(b) $PSNR_Y$



(c) $PSNR_{YUV}$



(d) $PSNR_Y$

Figure 6.6: Average PSNR value for different compression ratio on tested light field images *top* results for synthetic data. *bottom* results for real-world data.

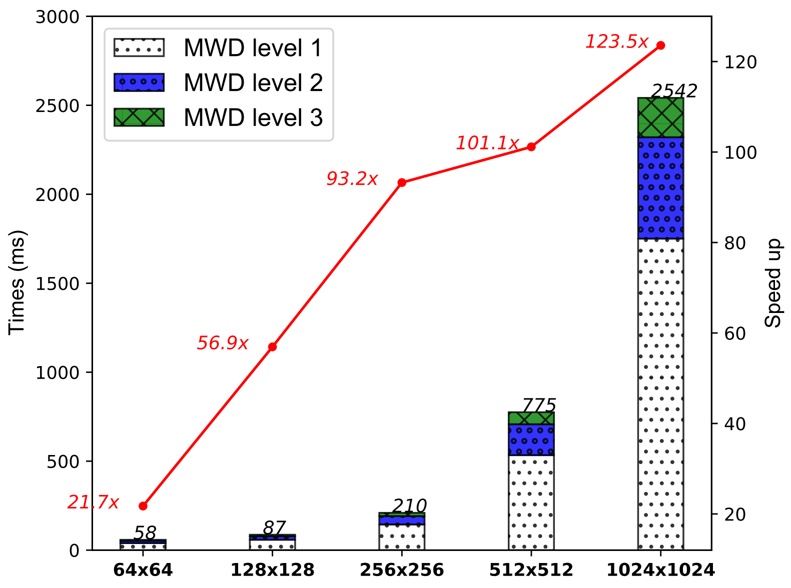


Figure 6.7: Timing evaluation of accelerated MWD in comparison to CPU implementation.

For disparity estimation, the algorithm and its implementation discussed in Chapter 2 and 3 are employed. Figure 6.5 demonstrates the estimated disparity map of 3 light field images used in our experiment. The disparity maps are smooth but still reserve sharp edges at depth discontinuing area. It is because of the intrinsic sub-pixel precision provided by the continuous formation and the sub-quadratic regularization. We notice that it is not necessary to use the finest level disparity maps for MWD. The performance loss is quite small when the down-sampled and quantized versions of disparity maps are used. In this experiment, we down-sampled the estimated disparity maps by a factor of two and quantized them to 2 bits representing three pixel shift levels -1,0, and 1. The modified disparity maps are encoded into the bit stream. They are bi-cubic up-scaled before being applied in the warping function.

The lossless compression results are listed in Table 6.1. The last two columns show the results of our coding framework using Haar and bi-orthogonal 5/3 wavelet kernel. The other columns report the compressed file size of JPEG2000, JPEG–LS and the original light field data size. From the table, it could be seen that the compressed file sizes produced by our algorithm are smaller than those compressed by JPEG–LS and JPEG2000. On average, it is about 25% and 5.5% better for synthetic data and 11% and 7.3 % better for real-world data compared to JPEG–LS and JPEG2000 respectively. This is attributed to MWD which effectively removes redundant information across multiple sub-aperture images and results in high-pass sub-band views. From Table 6.1, it also appears that the 5/3 wavelet kernel outperforms the Haar kernel for all of the test images.

For lossy compression, we set JPEG2000 encoder in our *Sub-band Compression* module to lossy mode. All the sub-band views are encoded with the same compression ratio. The disparity map is however always compressed losslessly. The compression ratios set for this test are 20,40,100,200. Figure 6.6 compares the average PSNR calculated on YUV color space. We use the equation suggested from JPEGPleno call for proposal [134],[135] for this PSNR computation. From the figure, it is apparent that our coding framework with 5/3 wavelet kernel consistently provides a better compression quality when compared to JPEG2000 in both synthetic and real-world datasets. The gain compared to JPEG2000 increases at a lower compression ratio and is even better for the luminance component. It could also be seen from the figure that the compression performance when using Haar kernel is different between synthetic and real-world datasets. It possesses the same compression quality as with 5/3 kernel on the real-world dataset but worse quality in the case of synthetic data.

Figure 6.7 shows the timing evaluation of the GPU-accelerated MWD. In this experiment, 4D LF images have the directional resolution of 9×9 while its spatial resolution is varied from 64×64 to 1024×1024. The amount of pixel data, therefore, ranges from 0.33 Mpx to 85 Mpx. The CPU implementation of MWD is executed on Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz while its GPU-accelerated version is run on Nvidia GeForce GTX 1080 Ti. Besides the total running time of MWD, the processing time of each MWD level and the speed-up of GPU-based MWD are also reported, see Figure 6.7. Overall GPU-based MWD demonstrates a superior performance as compared to CPU-based MWD. The gain in using GPU for MWD becomes more and more significant when the spatial resolution of 4D LF increases. At 64×64, the speed-up is 21.7×. This number is doubled at 128×128 and reach 123.5× at 1024×1024.

6.4 Summary

In this chapter, a compression framework based on motion-compensated wavelet decomposition for light field images has been proposed. By employing the estimated disparity maps for warping sub-aperture images in a two-step wavelet lifting scheme, we are able to generate high-/low-pass sub-band views in which redundant information existing in the pixel domain is suppressed. As being the second most time-consuming task in the coding framework after disparity estimation, a GPU-accelerated version of MWD is highly expected. An efficient acceleration strategy with 2 OpenCL kernels are proposed for 1D MWD computation. Experimental results show that our proposed framework performs better than JPEG–LS and JPEG2000. In the case of lossless compression, our approach provides 5.5% and 7.3% better compression ratio compared to JPEG2000 and 25% and 11% better compression ratio compared to JPEG–LS for synthetic and real-world data respectively. In the case of lossy compression, the result of using 5/3 wavelet kernel shows consistently better PSNR values compared to JPEG2000. The GPU-accelerated MWD achieves a significant speed-up as compared to CPU-based MWD, ranging from 21.7× to 123.5× for spatial resolution from 64×64 to 1024×1024 respectively.

Chapter 7

Summary and Outlook

7.1 Summary

In this dissertation we studied important 4DLF processing tasks including disparity estimation, super-resolution, and compression. For each of processing tasks, the output quality and computation time are considered as the two key aspects of the processing performance. On top of the proposed computational approaches which aim for high quality output, GPU is employed as a realization platform in order to alleviate the huge demand on computational resources and achieve high processing speed.

In Chapter 2 we presented a 4DLF disparity estimation approach which combines the robustness of variational optimization framework and effectiveness of weighted median filtering for computing an accurate disparity solution. Following optic flow literature, a systematic notation for variational 4DLF disparity estimation is introduced. This allows us to simplify the construction of the optimization problem as well the development of the numerical computation. In our optimization problem, the data term assembled the brightness and gradient constancy assumptions and isotropic total variation is employed for smoothness term. While separate robustifications are applied to the two constancy assumptions, we proposed a joint penalization of perspective images to reduce the computational effort. For optimization, coarse-to-fine warping scheme was employed to improve the accuracy in the case of large displacement. A detail numerical computation applying Euler-Lagrange equation to construct a linear system of equations and the derivation of explicit iterative solutions are also discussed.

In Chapter 3 we discussed the realization of the numerical computation presented in Chapter 2 on GPUs. The computation was decomposed and rearranged into a set of computing tasks which are then realized as OpenCL kernels. To cope with the high-dimension structure of 4D LF data, two memory layouts for storing the 4D LF data were investigated. An extensive evaluation was conducted, in which the performance of our approach was analyzed and compared to related approaches.

In Chapter 4 we presented a GPU-accelerated computation framework for spatial super-resolution of 4DLF image. The super-resolution model is derived from statistical perspective assembling a joint $\ell^1 - \ell^2$ data fidelity term and weighted regularization term. We showed that ADMM can be applied for effectively solving the non-smooth convex optimizaton problem. Tackle the limitation of sparse matrix implementation by linear function approximation, the iterative solver is realized on GPUs, thereby achieving a significant speed up as compared to sequentially solving on CPUs. Validation of the proposed super-resolution model, performance evaluation as well as the comparison to related works are also presented.

In Chapter 5 we presented a deep learning-based approach for dealing with the three super-resolution tasks namely spatial-, angular-, and spatial-angular super-resolution. The approach consists of two processing stages, the first stage preliminarily up-samples the 4DLF into desired resolution. The second stage enhances the quality of pre-scaled 4DLF in a novel EPI volume refinement network. The evaluation demonstrates our superior performance as compared to the state-of-the-art approaches.

In Chapter 6 we presented a compression approach for encoding 4DLF images. By employing the estimated disparity maps for warping sub-aperture images in a two-step wavelet lifting scheme, we are able to generate high-/low-pass sub-band views in which redundant information existing in the pixel domain is suppressed. The coding framework is tested with both synthetic and real-world light field dataset. The experimental results show that our approach outperforms JPEG–LS and the direct application of JPEG2000 in both lossless and lossy compression scenarios. In addition, a significant gain in processing time is achieved by a fast parallel computation on GPUs as compared to CPU-based implementation.

7.2 Outlook

In this dissertation, the acceleration of the 4DLF image processing tasks is conducted by the means of GPU execution. GPU is a preferable platform due to its capability of parallel processing and floating point operations. In addition, it is designed for handling image-like data such as 4DLF images. Important operations of 4DLF image processing such as filtering, interpolation, warping benefit greatly from the texture cache available in GPU platform. Besides GPU, Field Programmable Gate Array (FPGA) is also a good choice for accelerating the 4DLF processing tasks with the potential to achieve much higher processing speed and much lower energy consumption. However, there are some challenges which need to be overcome when realizing the proposed approaches on FPGA platform. First, there is a limited capacity for floating point operation on FPGA devices. All computations should be carried out in the form of fixed-point operations. The bit size of data and its dynamic range need to be investigated and decided for each of the processing tasks. Second, each processing operation need its own custom-logic which is typically described in hardware description language (HDL), such as VHDL or Verilog. Consequently, FPGA implementation requires a lot more effort as compared to GPU implementation. Although high level synthesis (HLS) tools offer a practical way to simplify the implementation process, one needs to decide between HLS and HDL to balance between development effort and performance. Third, FPGA has limited amount of low-latency memory (i.e., block ram or distributed ram) which, in general, cannot afford a full-size 4DLF image. 4DLF is therefore need to be kept in external memory. A proper accessing and buffering scheme is required for optimizing the 4DLF processing performance. Alleviating the mentioned challenges to bring the proposed approaches on the FPGA platform is listed as our future works.

Bibliography

[1] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. *Comput. Model. Vis. Process.*, pages 3–20, 1991.

[2] Marc Levoy and Pat Hanrahan. Light field rendering. In *ACM Proc. 23rd Annu. Conf. Comput. Graph. Interact. Tech.*, pages 31–42, 1996.

[3] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. *SIGGRAPH,*, pages 43–54, 1996.

[4] Katrin Honauer, Ole Johannsen, Daniel Kondermann, and Bastian Goldluecke. A dataset and evaluation methodology for depth estimation on 4d light fields. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 10113 LNCS:19–34, 2017.

[5] Marc Levoy. Light fields and computational imaging. *Computer (Long. Beach. Calif).*, (8):46–55, 2006.

[6] Juan Casavílca Silva, Muhammad Saadi, Lunchakorn Wuttisittikulkij, Davi Ribeiro Militani, Renata Lopes Rosa, Demóstenes Zegarra Rodríguez, and Sattam Al Otaibi. Light-field imaging reconstruction using deep learning enabling intelligent autonomous transportation system. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[7] Ryan S Overbeck, Daniel Erickson, Daniel Evangelakos, Matt Pharr, and Paul Debevec. A system for acquiring, processing, and rendering panoramic light field stills for virtual reality. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018.

[8] Lixia Ni, Zhenxing Li, Haifeng Li, and Xu Liu. 360-degree large-scale multiprojection light-field 3d display system. *Applied optics*, 57(8):1817–1823, 2018.

[9] Bennett Wilburn, Neel Joshi, Vaibhav Vaish, Eino-Ville Talvala, Emilio Antunez, Adam Barth, Andrew Adams, Mark Horowitz, and Marc Levoy. High performance imaging using large camera arrays. In *ACM Trans. Graph.*, volume 24, pages 765–776. ACM, 2005.

[10] Jonas Unger, Andreas Wenger, Tim Hawkins, Andrew Gardner, and Paul Debevec. Capturing and rendering with incident light fields. Technical report, Inst. Creative Tech., Univ. Southern California, 2003.

[11] Edward H Adelson and John Y A Wang. Single lens stereo with a plenoptic camera. *IEEE Trans. Pattern Anal. Mach. Intell.*, (2):99–106, 1992.

[12] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, Pat Hanrahan, and Others. Light field photography with a hand-held plenoptic camera. *Comput. Sci. Tech. Rep. CSTR*, 2(11):1–11, 2005.

[13] Andrew Lumsdaine and Todor Georgiev. The focused plenoptic camera. In *IEEE Int. Conf. Image Process.*, pages 1–8. IEEE, 2009.

[14] Sven Wanner, Stephan Meister, and Bastian Goldlucke. Datasets and benchmarks for densely sampled 4d light fields. In *VMV 2013 Vis. Model. Vis.*, pages 225–226, 2013.

[15] Jinglei Shi, Xiaoran Jiang, and Christine Guillemot. A framework for learning depth from a flexible subset of dense and sparse light field views. *IEEE Transactions on Image Processing*, pages 5867–5880, June 2019.

[16] Stanford lytro light field archive. Accessed: Feb. 2020.

[17] Lytro illum light field dataset. Accessed: Feb. 2020.

[18] Martin Rerabek and Touradj Ebrahimi. New light field image dataset. In *8th International Conference on Quality of Multimedia Experience (QoMEX)*, number CONF, 2016.

[19] Light fields from the lego gantry. Accessed: Feb. 2020.

[20] Stefan Heber, Rene Ranftl, and Thomas Pock. Variational shape from light field. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 8081 LNCS:66–79, 2013.

[21] Trung-Hieu Tran, Gasim Mammadov, and Sven Simon. Gvld: A fast and accurate gpu-based variational light-field disparity estimation approach. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(7):2562–2574, 2021.

[22] M. Shahzeb Khan Gul and Bahadir K. Gunturk. Spatial and angular resolution enhancement of light fields using convolutional neural networks. *IEEE Trans. Image Process.*, 27(5):2146–2159, 2018.

[23] In Kyu Park, Kyoung Mu Lee, and Others. Robust light field depth estimation using occlusion-noise aware data costs. *Trans. Pattern Anal. Mach. Intell.*, 40(10):2484–2497, 2018.

[24] Sven Wanner and Bastian Goldluecke. Variational light field analysis for disparity estimation and super-resolution. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(3):606–619, 2014.

[25] Changha Shin, Hae-Gon Jeon, Youngjin Yoon, In So Kweon, and Seon Joo Kim. Epinet: A fully-convolutional neural network using epipolar geometry for depth from light field images. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 4748–4757, 2018.

[26] Stefan Heber, Wei Yu, and Thomas Pock. Neural epi-volume networks for shape from light field. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 2271–2279, 2017.

[27] Trung-Hieu Tran, Jan Berberich, and Sven Simon. 3dvsr: 3d epi volume-based approach for angular and spatial light field image super-resolution. *Signal Processing*, 192:108373, 2022.

[28] Hae-Gon Jeon, Jaesik Park, Gyeongmin Choe, Jinsun Park, Yunsu Bok, Yu-Wing Tai, and In So Kweon. Accurate depth map estimation from a lenslet light field camera. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1547–1555, 2015.

[29] Michael W Tao, Sunil Hadap, Jitendra Malik, and Ravi Ramamoorthi. Depth from combining defocus and correspondence using light-field cameras. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 673–680, 2013.

[30] Ting-Chun Wang, Alexei A Efros, and Ravi Ramamoorthi. Occlusion-aware depth estimation using light-field cameras. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 3487–3495, 2015.

[31] Alessandro Neri, Marco Carli, and Federica Battisti. A multi-resolution approach to depth field estimation in dense image arrays. In *IEEE Int. Conf. Image Process.*, pages 3358–3362. IEEE, 2015.

[32] Lipeng Si and Qing Wang. Dense depth-map estimation and geometry inference from light fields via global optimization. In *Asian Conf. Comput. Vis.*, pages 83–98. Springer, 2016.

[33] Yuriy Anisimov and Didier Stricker. Fast and efficient depth map estimation from light fields. In *Int. Conf. 3D Vis.*, pages 337–346. IEEE, IEEE, oct 2017.

[34] Jae Young Lee and Rae-Hong Park. Depth estimation from light field by accumulating binary maps based on foreground–background separation. *IEEE J. Sel. Top. Signal Process.*, 11(7):955–964, 2017.

[35] Michael Strecke, Anna Alperovich, and Bastian Goldluecke. Accurate depth and normal maps from occlusion-aware focal stack symmetry. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 2814–2822, 2017.

[36] 4d light field benchmark. Accessed: Oct. 2019.

[37] Yaoxiang Luo, Wenhui Zhou, Junpeng Fang, Linkai Liang, Hua Zhang, and Guojun Dai. Epi-patch based convolutional neural network for depth estimation on 4d light field. In *Proc. Int. Conf. Neural Information Processing*, pages 642–652, 2017.

[38] Stefan Heber, Wei Yu, and Thomas Pock. U-shaped networks for shape from light field. *Br. Mach. Vis. Conf. 2016, BMVC 2016*, 2016-Septe(1):37.1–37.12, 2016.

[39] Haoxin Ma, Haotian Li, Zhiwen Qian, Shengxian Shi, and Tingting Mu. Vommanet: an end-to-end network for disparity estimation from reflective and texture-less light field images. *arXiv Prepr. arXiv1811.07124*, 2018.

[40] Titus Leistner, Hendrik Schilling, Radek Mackowiak, Stefan Gumhold, and Carsten Rother. Learning to think outside the box: Wide-baseline light field depth estimation with epi-shift. In *Int. Conf. 3D Vis.*, 2019.

[41] Yuttakon Yuttakonkit, Shinya Takamaeda-Yamazaki, and Yasuhiko Nakashima. Performance optimization of light-field applications on gpu. *IEICE Trans. Inf. Syst.*, 99(12):3072–3081, 2016.

[42] Yanwen Qin, Xin Jin, and Qionghai Dai. Gpu-based depth estimation for light field images. In *2017 Int. Symp. Intell. Signal Process. Commun. Syst.*, volume 6, pages 640–645. IEEE, nov 2017.

[43] Andre Ivan, In Kyu Park, and Others. Light field depth estimation on off-the-shelf mobile gpu. In *IEEE Conf. Comput. Vis. Pattern Recognit. Work.*, pages 634–643, 2018.

[44] K. Mitra and A. Veeraraghavan. Light field denoising, light field superresolution and stereo camera based refocussing using a gmm light field patch prior. In *2012 IEEE*

Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pages 22–28, June 2012.

[45] T. E. Bishop and P. Favaro. The light field camera: Extended depth of field, aliasing, and superresolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):972–986, May 2012.

[46] M. Alain and A. Smolic. Light field super-resolution via lfbm5d sparse coding. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 2501–2505, Oct 2018.

[47] M. Rossi, M. E. Gheche, and P. Frossard. A nonsmooth graph-based approach to light field super-resolution. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 2590–2594, Oct 2018.

[48] Trung Hieu Tran, Gasim Mammadov, Kaicong Sun, and Sven Simon. Gpu-accelerated light-field image super-resolution. In *Proc. - 2018 Int. Conf. Adv. Comput. Appl. ACOMP 2018*, pages 7–13. IEEE, 2018.

[49] M. Rossi and P. Frossard. Geometry-consistent light field super-resolution via graph-based regularization. *IEEE Transactions on Image Processing*, 27(9):4207–4218, Sep. 2018.

[50] Youngjin Yoon, Hae-Gon Jeon, Donggeun Yoo, Joon-Young Lee, and In So Kweon. Learning a deep convolutional network for light-field image super-resolution. In *Conf. Comput. Vis. Work.*, 2015.

[51] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*, pages 184–199. Springer, 2014.

[52] Youngjin Yoon, Hae Gon Jeon, Donggeun Yoo, Joon Young Lee, and In So Kweon. Light-field image super-resolution using convolutional neural network. *IEEE Signal Process. Lett.*, 24(6):848–852, jun 2017.

[53] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2016)*, 35(6), 2016.

[54] Reuben A. Farrugia, Christian Galea, and Christine Guillemot. Super resolution of light field images using linear subspace projection of patch-volumes. *IEEE J. Sel. Top. Signal Process.*, 11(7):1058–1071, 2017.

[55] Hanzhi Fan, Dong Liu, Zhiwei Xiong, and Feng Wu. Two-stage convolutional neural network for light field super-resolution. In *Int. Conf. Image Process.*, pages 1167–1171. IEEE, sep 2017.

[56] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *2016 IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1646–1654. IEEE, jun 2016.

[57] Yan Yuan, Ziqi Cao, and Lijuan Su. Light-field image superresolution using a combined deep cnn based on epi. *IEEE Signal Process. Lett.*, 25(9):1359–1363, 2018.

[58] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *IEEE Conf. Comput. Vis. Pattern Recognit. Work.*, jul 2017.

[59] Yunlong Wang, Fei Liu, Kunbo Zhang, Guangqi Hou, Zhenan Sun, and Tieniu Tan. Lfnet: A novel bidirectional recurrent convolutional neural network for light-field image super-resolution. *IEEE Trans. Image Process.*, 27(9):4274–4286, 2018.

[60] Henry Wing Fung Yeung, Junhui Hou, Xiaoming Chen, Jie Chen, Zhibo Chen, and Yuk Ying Chung. Light field spatial super-resolution using deep efficient spatial-angular separable convolution. *IEEE Transactions on Image Processing*, 28(5):2319–2330, 2018.

[61] Shuo Zhang, Youfang Lin, and Hao Sheng. Residual networks for light field image super-resolution. *IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 11046–11055, 2019.

[62] Simon Baker and Takeo Kanade. Limits on super-resolution and how to break them. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(9):1167–1183, 2002.

[63] S. Farsiu, M. D. Robinson, M. Elad, and P. Milanfar. Fast and robust multiframe super resolution. *IEEE Transactions on Image Processing*, 13(10):1327–1344, Oct 2004.

[64] Kaicong Sun, Trung-Hieu Tran, Jajnabalkya Guhathakurta, and Sven Simon. Fl-misr: fast large-scale multi-image super-resolution for computed tomography based on multi-gpu acceleration. *Journal of Real-Time Image Processing*, pages 1–14, 2021.

[65] R. Monteiro, L. Lucas, C. Conti, P. Nunes, N. Rodrigues, S. Faria, C. Pagliari, E. da Silva, and L. Soares. Light field hevc-based image coding using locally linear embedding and self-similarity compensated prediction. In *2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pages 1–4, July 2016.

[66] Y. Li, R. Olsson, and M. Sjöström. Compression of unfocused plenoptic images using a displacement intra prediction. In *2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pages 1–4, July 2016.

[67] C. Conti, P. Nunes, and L. D. Soares. Hevc-based light field image coding with bi-predicted self-similarity compensation. In *2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pages 1–4, July 2016.

[68] D. Liu, L. Wang, L. Li, Zhiwei Xiong, Feng Wu, and Wenjun Zeng. Pseudo-sequenli2016compressionce-based light field image compression. In *2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pages 1–4, July 2016.

[69] C. Perra and P. Assuncao. High efficiency coding of light field images based on tiling and pseudo-temporal data arrangement. In *2016 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pages 1–4, July 2016.

[70] Petri Helin, Pekka Astola, Bhaskar Rao, and Ioan Tabus. Sparse modelling and predictive coding of subaperture images for lossless plenoptic image compression. In *2016 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, pages 1–4. IEEE, jul 2016.

[71] Donald G Dansereau, Oscar Pizarro, and Stefan B Williams. Decoding, calibration and rectification for lenselet-based plenoptic cameras. In *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, pages 1027–1034, 2013.

[72] Trung-Hieu Tran, Zhe Wang, and Sven Simon. Variational disparity estimation framework for plenoptic images. In *IEEE Int. Conf. Multimed. Expo*, pages 1189–1194, 2017.

[73] Masatoshi Okutomi and Takeo Kanade. A multiple-baseline stereo. *IEEE Transactions on pattern analysis and machine intelligence*, 15(4):353–363, 1993.

[74] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *Eur. Conf. Comput. Vis.*, pages 25–36. Springer, 2004.

[75] Andrés Bruhn, Joachim Weickert, Timo Kohlberger, and Christoph Schnörr. A multigrid platform for real-time motion computation with discontinuity-preserving variational methods. *International Journal of Computer Vision*, 70(3):257–277, 2006.

[76] Rene Ranftl, Stefan Gehrig, Thomas Pock, and Horst Bischof. Pushing the limits of stereo using variational stereo estimation. In *2012 IEEE Intelligent Vehicles Symposium*, number 1, pages 401–407. IEEE, jun 2012.

[77] Andrés Bruhn and Joachim Weickert. Towards ultimate motion estimation: Combining highest accuracy with real-time performance. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 749–755. IEEE, 2005.

[78] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr. Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International journal of computer vision*, 61(3):211–231, 2005.

[79] Gunnar Farneback. Very high accuracy velocity estimation using orientation tensors, parametric motion, and simultaneous segmentation of the motion field. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 1, pages 171–177. IEEE, 2001.

[80] Tony F Chan and Pep Mulet. On the convergence of the lagged diffusivity fixed point method in total variation image restoration. *SIAM journal on numerical analysis*, 36(2):354–367, 1999.

[81] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 2432–2439. IEEE, 2010.

[82] Ziyang Ma, Kaiming He, Yichen Wei, Jian Sun, and Enhua Wu. Constant time weighted median filtering for stereo matching and beyond. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 49–56, 2013.

[83] Yingying Li and Stanley Osher. A new median formula with applications to pde based denoising. *Communications in Mathematical Sciences*, 7(3):741–753, 2009.

[84] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(6):1397–1409, 2012.

[85] Peter Sand and Seth Teller. Particle video: Long-range motion estimation using point trajectories. *Int. J. Comput. Vis.*, 80(1):72, 2008.

[86] Henning Zimmer, Andrés Bruhn, and Joachim Weickert. Optic flow in harmony. *International Journal of Computer Vision*, 93(3):368–388, 2011.

[87] Andrés Bruhn. Correspondence problems in computer vision. Univesity of Stuttgart, Institute for Visualization and Interactive Systems. Lecture Notes, April 2016.

[88] Sven Wanner and Bastian Goldluecke. Globally consistent depth labeling of 4d light fields. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 41–48. IEEE, 2012.

[89] Stefan Heber and Thomas Pock. Shape from light field meets robust pca. In *Eur. Conf. Comput. Vis.*, pages 751–767. Springer, 2014.

[90] Nvidia Corporation. OpenCL Programming Guide for the CUDA Architecture version 4.2. Technical report, 2012.

[91] Katrin Honauer, Ole Johannsen, Daniel Kondermann, and Bastian Goldluecke. A dataset and evaluation methodology for depth estimation on 4d light fields. In *Asian Conf. Comput. Vis.*, pages 19–34. Springer, 2016.

[92] 4d light-field benchmark tools. Accessed: Oct. 2019.

[93] Yuriy Anisimov, Oliver Wasenmüller, and Didier Stricker. Rapid light field depth estimation with semi-global matching. jul 2019.

[94] Chao-Tsung Huang. Robust pseudo random fields for light-field stereo matching. In *IEEE Int. Conf. Comput. Vis.*, volume 2017-Octob, pages 11–19. IEEE, oct 2017.

[95] Ole Johannsen, Antonin Sulc, and Bastian Goldluecke. What sparse light field coding reveals about scene structure. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3262–3270, 2016.

[96] Hae-Gon Jeon, Jaesik Park, Gyeongmin Choe, Jinsun Park, Yunsu Bok, Yu-Wing Tai, and In So Kweon. Depth from a light field image with learning-based matching costs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(2):297–310, 2019.

[97] Shuo Zhang, Hao Sheng, Chao Li, Jun Zhang, and Zhang Xiong. Robust depth estimation for light field via spinning parallelogram operator. *Comput. Vis. Image Underst.*, 145:148–159, 2016.

[98] Shuhang Gu, Andreas Lugmayr, Martin Danelljan, Manuel Fritsche, Julien Lamour, and Radu Timofte. Div8k: Diverse 8k resolution image dataset. In *Int. Conf. on Comp. Vis. Work.*, pages 3512–3516, 2019.

[99] Trung-Hieu Tran, Kaicong Sun, and Sven Simon. A gpu-accelerated light-field super-resolution framework based on mixed noise model and weighted regularization. *Journal of Real-Time Image Processing*, 19(5):893–910, 2022.

[100] R. Farrugia and C. Guillemot. Light field super-resolution using a low-rank prior and deep convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2019.

[101] Kaicong Sun, Trung-Hieu Tran, Roman Krawtschenko, and Sven Simon. Multi-frame super-resolution reconstruction based on mixed poisson–gaussian noise. *Signal Processing: Image Communication*, 82:115736, 2020.

[102] Paul Rodríguez. Total variation regularization algorithms for images corrupted with different noise models: A review. *J. Electr. Comput. Eng.*, 2013(1), 2013.

[103] Mila Nikolova. A variational approach to remove outliers and impulse noise. *Journal of Mathematical Imaging and Vision*, 20(1):99–120, 2004.

[104] Tongtong Jia, Yuying Shi, Yonggui Zhu, and Lei Wang. An image restoration model combining mixed l1/l2 fidelity terms. *J. Vis. Commun. Image Represent.*, 38:461–473, 2016.

[105] M. Hakim, A. Ghazdali, and A. Laghrib. A multi-frame super-resolution based on new variational data fidelity term. *Applied Mathematical Modelling*, 87:446–467, 2020.

[106] Martin Pleschberger, Stefan Schrunner, and Jürgen Pilz. An Explicit Solution for Image Restoration using Markov Random Fields. *Journal of Signal Processing Systems*, 92(2):257–267, 2020.

[107] Federica Sciacchitano, Y Dong, and PC Hansen. *Image reconstruction under non-Gaussian noise*. PhD thesis, PhD thesis. Technical University of Denmark, 2017.

[108] Stuart Geman and Donald Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984.

[109] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[110] Stefan Kindermann, Stanley Osher, and Peter W Jones. Deblurring and denoising of images by nonlocal functionals. *Multiscale Modeling & Simulation*, 4(4):1091–1115, 2005.

[111] Guy Gilboa and Stanley Osher. Nonlocal operators with applications to image processing. *Multiscale Modeling and Simulation*, 7(3):1005–1028, 2008.

[112] K. Sun and S. Simon. Bilateral spectrum weighted total variation for noisy-image super-resolution and image denoising. *IEEE Trans. Signal Process.*, 69:6329–6341, 2021.

[113] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

[114] Naum Zuselevich Shor. *Minimization methods for non-differentiable functions*, volume 3. Springer Science & Business Media, 2012.

[115] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[116] W. J. Kammerer and M. Z. Nashed. On the Convergence of the Conjugate Gradient Method for Singular Linear Operator Equations. *SIAM Journal on Numerical Analysis*, 9(1):165–181, mar 1972.

[117] Markus Unger, Thomas Pock, Manuel Werlberger, and Horst Bischof. A convex approach for variational super-resolution. In *Joint pattern recognition symposium*, pages 313–322. Springer, 2010.

[118] Mantang Guo, Junhui Hou, Jing Jin, Jie Chen, and Lap-Pui Chau. Deep spatial-angular regularization for light field imaging, denoising, and super-resolution. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (01):1–1, 2021.

[119] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 2261–2269, 2017.

[120] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *ECCV*, 2018.

[121] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[122] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Int. Conf. Learning Representations*, May 2019.

[123] M. Haris, G. Shakhnarovich, and N. Ukita. Deep back-projection networks for super-resolution. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1664–1673, June 2018.

[124] Yunlong Wang, Fei Liu, Zilei Wang, Guangqi Hou, Zhenan Sun, and Tieniu Tan. End-to-end view synthesis for light field imaging with pseudo 4dcnn. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[125] G. Wu, Y. Liu, Q. Dai, and T. Chai. Learning sheared epi structure for light field reconstruction. *IEEE Transactions on Image Processing*, 28(7):3261–3273, 2019.

[126] G. Wu, Y. Liu, L. Fang, Q. Dai, and T. Chai. Light field reconstruction using convolutional network on epi and extended applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(7):1681–1694, 2019.

[127] Trung-hieu Tran, Yousef Baroud, Zhe Wang, Sven Simon, and David Taubman. Light-field image compression based on variational disparity estimation and motion-compensated wavelet decomposition. In *IEEE Int. Conf. Image Process.*, pages 3260–3264, 2017.

[128] Raytrix| 3d light-field machine vision. Accessed: Jul. 2022.

[129] JPEG Pleno Database: EPFL Light-field data set. http://jpeg.org/plenodb/lf/epfl/.

[130] Wim Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM Journal on Mathematical Analysis*, 29(2):511–546, 1998.

[131] Jiri Matela, Vít Rusňák, and Petr Holub. Efficient jpeg2000 ebcot context modeling for massively parallel architectures. In *2011 Data Compression Conference*, pages 423–432. IEEE, 2011.

[132] Jiří Matela, Martin Šrom, and Petr Holub. Low gpu occupancy approach to fast arithmetic coding in jpeg2000. In *International Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, pages 136–145. Springer, 2011.

[133] Miłosz Ciżnicki, Michał Kierzynka, Piotr Kopta, Krzysztof Kurowski, and Paweł Gepner. Benchmarking jpeg 2000 implementations on modern cpu and gpu architectures. *Journal of Computational Science*, 5(2):90–98, 2014.

[134] T. Ebrahimi, S. Foessel, F. Pereira, and P. Schelkens. Jpeg pleno: Toward an efficient representation of visual reality. *IEEE MultiMedia*, 23(4):14–20, Oct 2016.

[135] Workplan and Specs of JPEG Pleno. http://jpeg.org/jpegpleno/workplan.html.

List of Publications

Journal Articles

1. T.H. Tran, K. Sun, and S. Simon, “A GPU-Accelerated Light-field Super-resolution Framework Based on Mixed Noise Model and Weighted Regularization,” Journal of Real-time Image Processing, 19(5), pp. 893-910, 2022.
2. T.H. Tran, J. Berberich, and S. Simon, “3DVSR: 3D EPI Volume-based Approach for Angular and Spatial Light-field Image Super-Resolution,” Signal Processing, vol. 192, p. 108373, 2022.
3. T.H. Tran, G. Mammadov, and S. Simon, “GVLD: A Fast and Accurate Gpu-based Variational Light-field Disparity Estimation Approach,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 31, no. 7, pp. 2562–2574, 2021.
4. K. Sun, T.H. Tran, J. Guhathakurta, and S. Simon, “FL-MISR: Fast Large-Scale Multi-Image Super-Resolution for Computed Tomography based on Multi-GPU Acceleration,” Journal of Real- Time Image Processing, 19(2), 331-344, 2021.
5. K. Sun, T.H. Tran, R. Krawtschenko, and S. Simon, “Multi-Frame Super-Resolution Reconstruction based on Mixed Poisson–Gaussian Noise,” Signal Processing: Image Communication, vol. 82, p. 115736, 2020.
6. Z. Wang, T.H. Tran, P.K. Muthappa and S. Simon, “A JND-based pixel-domain algorithm and hardware architecture for perceptual image coding,” Journal of Imaging, 5(5), p.50, 2019.
7. S.M. Najmabadi, T.H. Tran, S. Eissa, H.S. Tungal, and S. Simon, “An Architecture for Asymmetric Numeral Systems Entropy Decoder - a Comparison with a Canonical Huffman decoder,” Journal of Signal Processing Systems, 91(7), pp. 805-817, 2019.

Peer-reviewed Conference Papers

1. T.H. Tran, S. Simon, B. Chen, D. Russ, D. Claus, “A lightweight and robust VCSEL-based 3D-depth sensing approach for mobile application,” in Digital Optical Technologies 2021 (Vol. 11788, pp. 70-77). SPIE.
2. T.H. Tran, G. Mammadov, K. Sun, and S. Simon, “GPU-Accelerated Light-field Image Super- Resolution,” in International Conference on Advanced Computing and Applications (ACOMP), pp. 7–13. IEEE, 2018.
3. T.H. Tran, Z. Wang, and S. Simon, “Variational Disparity Estimation Framework for Plenoptic Images,” In IEEE International Conference on Multimedia and Expo (ICME), pp. 1189-1194. IEEE, 2017. **Best PhD student paper award**
4. T.H. Tran, Y. Baroud, Z. Wang, S. Simon, and D. Taubman, “Light-field Image Compression Based on Variational Disparity Estimation and Motion-Compensated Wavelet Decomposition,” in IEEE International Conference Image Processing, pp. 3260–3264. IEEE, 2017.
5. S.M. Najmabadi, P. Pandit, T.H. Tran, and S. Simon, “A resource-efficient monitoring architecture for hardware accelerated self-adaptive online data stream compression,” In 2017 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), pp. 222-227. IEEE, 2017. **Best paper award**
6. S.M. Najmabadi, H.S. Tungal, T.H. Tran, and S. Simon, “Hardware-based architecture for asymmetric numeral systems entropy decoder,” In 2017 Conference on Design and Architectures for Signal and Image Processing (DASIP), pp. 1-6. IEEE, 2017.