JOINT DEMOSAICKING / RECTIFICATION OF FISHEYE CAMERA IMAGES USING MULTI-COLOR GRAPH LAPLACIAN REGULARIZATION

FENGBO LAN

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTERS OF APPLIED SCIENCE

GRADUATE PROGRAM IN ELECTRICAL AND COMPUTER ENGINEERING
YORK UNIVERSITY
TORONTO, ONTARIO

MAY 2021

Abstract

To compose one 360° image from multiple viewpoint images taken from different fisheye cameras on a rig for viewing on a head-mounted display (HMD), a conventional processing pipeline first performs demosaicking on each fisheye camera's Bayer-patterned grid, then translates demosaicked pixels from the camera grid to a rectified image grid. By performing two image interpolation steps in sequence, interpolation errors can accumulate, and acquisition noise in each captured pixel can pollute its neighbors, resulting in correlated noise. In this paper, a joint processing framework is proposed that performs demosaicking and grid-to-grid mapping simultaneously, thus limiting noise pollution to one interpolation. Specifically, a reverse mapping function is first obtained from a regular on-grid location in the rectified image to an irregular off-grid location in the camera's Bayer-patterned image. For each pair of adjacent pixels in the rectified grid, its gradient is estimated using the pair's neighboring pixel gradients in three colors in the Bayer-patterned grid. A similarity graph is constructed based on the estimated gradients, and pixels are interpolated in the rectified grid directly via graph Laplacian regularization (GLR). To establish ground truth for objective testing, a large dataset containing pairs of simulated images both in the fisheye camera grid and the rectified image grid is built.

Experiments show that the proposed joint demosaicking / rectification method outperforms competing schemes that execute demosaicking and rectification in sequence in both objective and subjective measures.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The advent of camera technologies means that images can now be captured in non-traditional formats with large fields-of-view for modern imaging applications. One example is fisheye camera such as Kandao Qoocam[1] or GoPro Max[2] that is capable of capturing an ultra-wide but irregularly-shaped field-of-view. An example fisheye camera image is shown in Fig. 1.1a. Multiple viewpoint images captured by different fisheye cameras on a rig [54, 99] can be stitched together using algorithms such as GIST [67] to compose a 360° image, which can be viewed subsequently in a head-mounted display (HMD) such as HTC Vive[3] or Oculus Rift[4] for an immersive viewing experience. The ability to observe ultra-realistic 3D scenes in HMD is essential for a wide variety of imaging applications, including virtual reality (VR) [99], augmented reality (AR), surveillance [60], and immersive communication [87] for remote learning

[1]https://www.kandaovr.com/qoocam-8k/

[2]https://gopro.com/en/us/shop/cameras/max/CHDHZ-201-master.html

[3]https://www.vive.com/

[4]https://www.oculus.com/rift/

and tele-surgery [92], etc.

However, captured images by fisheye cameras are often geometrically distorted because of the optical structures of fisheye camera lenses. As illustrated in Fig. 1.1b, straight lines on the fisheye camera images are visibly bent. Thus, *image rectification* [50] is necessary before projecting these captured images onto a physical display for visual consumption.

In general, rectification is a transformation process that realigns an image onto a target image plane by performing image interpolation. The interpolation position can be computed with a pre-computed camera calibration matrix [120], which includes distortion information of the lenses. In recent literature, some rectification methods without calibration procedures are proposed, by performing rectification using the distorted information from the input fisheye camera images. For example, line-based methods assume that straight lines are projected to circular arcs in the image plane caused by radial lens distortion, such that the distortion parameters can be estimated through arc fitting [109, 14, 100].

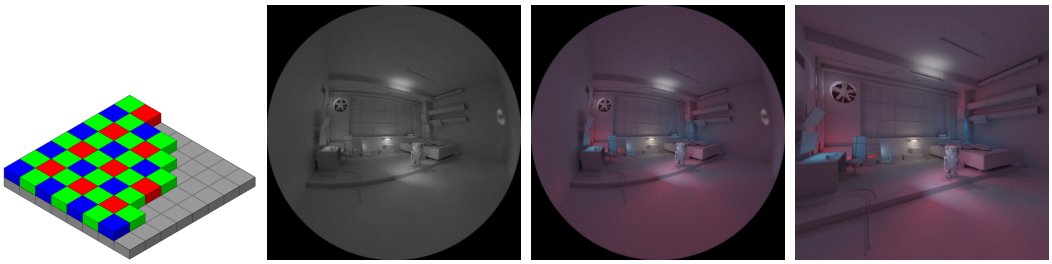
In a conventional image processing pipeline, before applying rectification on fisheye camera images (*e.g.*, from Fig. 1.2c to Fig. 1.2d), *image demosaicking* (*e.g.*, from Fig. 1.2b to Fig. 1.2c) will be performed first. This is because, using a Bayer-patterned array, as shown in Fig. 1.2a, the camera sensor only captures one color channel at each pixel location, which results in a Bayer-patterned image as shown in Fig. 1.2b. Thus, to construct a color image like Fig. 1.2c with all three color components (*i.e.*, red, blue and green) at each pixel location, image demosaicking needs to be performed to interpolate missing color components.

(a) (b)

Figure 1.1: Example of fisheye camera images.

Recent popular methods for demosaicking perform image interpolation that fills in missing color information using a local neighborhood of observed pixels [12, 117]. In comparison, non-local methods [11, 49], such as *non-local mean* (NLM) [118], tend to have better demosaicking performance, but they are limited by high complexity and long running time.

Image demosaicking and rectification are performed separately in a conventional image processing pipeline. However, during the capturing process, as light photons are collected at each pixel location to compute an intensity value, additive noise is typically observed, which corrupts the intensity computation. Performing demosaicking and rectification in sequence means that image interpolation is executed *twice*, where at each stage, errors can accumulate, and acquisition noise at the captured pixels can smear neighbors, resulting in correlated noise.



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

Figure 1.2: Processing pipeline for fisheye camera images. (a) Bayer filter array on the camera sensor, which captures one color component at each pixel location. (b) Bayer-pattern image, which is noisy in the real scene. The noise is inevitably introduced during the capturing process. (c) Demosaicked color image, which has three color channels on each pixel location. (d) Rectified image.

The removal of correlated noise is more challenging than removing uncorrelated ones [46, 88]. In fact, *independent and identically distributed* (iid) additive noise is assumed in most previous notable image denoising works, such as bilateral filtering [85], NLM [10], and *block-matching and 3D filtering* (BM3D) [25]. A denoising engine developed with the assumption of uncorrelated additive noise, such as *additive white Gaussian noise* (AWGN), tends to perform poorly when applied to images with correlated noise [122].

To alleviate this problem, in this thesis a joint demosaicking and rectification scheme is proposed for fisheye camera images. For example, Fig. 1.2d will be obtained by interpolating Fig. 1.2b directly. This means that image interpolation is performed only *once*, which limits the effect of noise pollution during only one interpolation step and potentially leads to better performance.

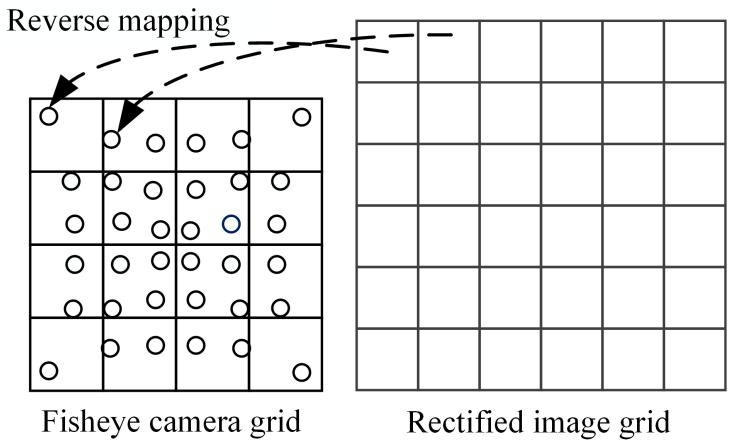


Figure 1.3: An illustration of mapping in rectification.

In this thesis, the joint demosaicking problem is formulated as an inverse imaging problem using *graph signal processing* (GSP) based regularization for pixel interpolation. GSP [83, 95, 21] studies the processing of signals residing on graphs. In recent years, GSP has been successfully applied to many image processing tasks, such as image restoration [3, 115, 70, 84, 19], image compression [39, 40, 56, 17, 93], and image segmentation [81, 86, 64, 8]. GSP is suitable for the joint demosaicking / rectification problem, because the problem requires processing of non-integer signal samples that are not conventional 2D images on a regular 2D grid, and a graph-based approach can flexibly handle irregular sample placements. To illustrate this, consider the rectification process involving grid-to-grid mapping in Fig. 1.3. Two "on-grid" pixels in the rectified image grid in the right map to two "off-grid" pixel locations in the fisheye camera grid. The joint demosaicking / rectification problem is the problem of interpolating directly these off-grid pixels translated from the rectified image grid using only sparse on-grid color pixels in the fisheye camera grid. A cleverly constructed graph can elegantly account for pairwise similarity information for

neighboring pixel pair in the rectified grid and the fisheye camera grid, as well as pixel pairs across the two grids. Having constructed an appropriate graph connected all relevant pixels, a target pixel patch can then be computed via graph spectral filtering.

This thesis is structured as follows. In Chapter 2, a review on related works in model-based image restoration, basic graph signal processing (GSP) concepts, applications of GSP in image restoration, and joint demosaicking problems is provided. The proposed joint demosaicking and rectification algorithm is described in Chapter 3. In Chapter 4, the construction of an image dataset for experiments is discussed. Experimental results are presented and discussed in Chapter 5. Finally, a conclusion is provided in Chapter 6.

Chapter 2

Related Works

In this chapter, related works in model-based image restoration problems are first discussed in Sec. 2.1. A brief introduction of *Graph Signal Processing* (GSP) concepts is provided in Sec. 2.2. Literature in graph-based image processing techniques is reviewed in Sec. 2.3. In Sec. 2.4, an overview on joint demosaicking problems is provided.

2.1 Model-based Image Restoration

In this thesis, joint demosaicking and rectification is formulated as an inverse imaging optimization problem. Optimization formulation is common in the literature when addressing image restoration tasks, such as denoising [84] and deblurring [3], using a model-based approach. To facilitate understanding, a linear image formation model is first introduced in Sec. 2.1.1. Next, a *Maximum a Posteriori (MAP)* formulation based on the model is introduced in Sec. 2.1.2.

2.1.1 Image Formation Model

An image (or image patch) that has been corrupted by additive noise, blurring, and/or down-sampling can be described by the following linear model [84, 57, 3, 36, 82, 113, 34, 1]:

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n}, \tag{2.1}$$

where $\mathbf{y} \in \mathbb{R}^m$ and $\mathbf{x} \in \mathbb{R}^n$ are the corrupted and original versions of the image in vector form. $\mathbf{H} \in \mathbb{R}^{m \times n}$ is a matrix that can low-pass filter (blur) and/or down-sample the image, and $\mathbf{n} \in \mathbb{R}^m$ is an additive noise, typically with zero mean and a known probability distribution. $\mathbf{H}$ can take on different definitions depending on the intended applications.

For example, in *image denoising* [18, 107], the goal is to restore the original noise-free image $\mathbf{x}$ given only a noisy observation $\mathbf{y}$. $\mathbf{H}$ is an identity matrix in this case. Additive noise $\mathbf{n}$ is often assumed to be independent and identically distributed (iid). For example, *additive white Gaussian noise* (AWGN) with a known noise variance is assumed in several notable works in the image denoising literature [25, 102, 10]. However, in practice, the noise on images captured using real camera sensors is a mixture of multiple noise sources following different distributions. Hence, to more accurately simulate sensor noise, $\mathbf{n}$ is modeled using more complex distributions in recent works, such as signal-dependent models [72, 71, 103].

If $\mathbf{H}$ is a matrix performing both downsampling and a low-pass filtering operation that mimics the *point spread function* (PSF) [111], *e.g.*, bilinear interpolation matrix with dimension $m < n$, then Eq. (2.1) can be interpreted as an *image super-resolution* problem [78]. Noise $\mathbf{n}$ is often assumed to be zero

in this case. Thus, $\mathbf{y}$ is a low-resolution image that is observable, and $\mathbf{x}$ is the corresponding target high-resolution image to be reconstructed.

When $\mathbf{H}$ is a blurring operator, Eq. (2.1) can be interpreted as an *image deblurring* problem [90]. Specifically, $\mathbf{H}$ is a square matrix associated with a blurring kernel, which typically is also a low-pass filter. If $\mathbf{H}$ is unknown, it is estimated during the optimization process. The deblurring problem without knowing $\mathbf{H}$ *a priori* is also called *blind image deblurring* [3].

Image interpolation [19, 89] and *image inpainting* [5, 94] are related problems, where $\mathbf{H} \in \{0,1\}^{m \times n}$ is a *sampling matrix* that identifies m observable pixels out of n, where $m < n$. (Each row of a sampling matrix contains a single 1 and the rest of the entries are 0's.) Typically, no additive noise is considered, *i.e.*, $\mathbf{n} = \mathbf{0}$. The difference between image inpainting and interpolation is that unknown pixels are collocated in the same spatial region in image inpainting, while unknown pixels are more evenly distributed in image interpolation. This means that methods such as *partial differential equations* (PDEs) [13, 38] that perform well for filling small holes in image interpolation tend not to work for image inpainting.

In practice, an observed image may be a result of multiple corruptions, making the image restoration problem even more challenging. For example, in [114] a blurred image to be recovered is also assumed to be noise-corrupted. This is true for an image captured at night by a handheld device like a mobile phone, where the image tends to be blurry due to long exposure time and hand movement, and noisy due to the low-light environment and the typical high ISO setting. Similarly, a multi-frame super resolution degradation model is discussed in [36], where the authors assume that the observable images are

noisy, blurred and have low-resolution.

2.1.2 MAP Formulation for Restoration

Given the image formation model in (2.1), recovering the original signal $\mathbf{x}$ can be formulated as a *Maximum a Posteriori* (MAP) problem; *i.e.*, find signal $\mathbf{x}$ that maximizes the posterior probability $P(\mathbf{x}|\mathbf{y})$ given observation $\mathbf{y}$:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} P(\mathbf{x} \,|\, \mathbf{y}). \tag{2.2}$$

Using Bayes' rule, we can write

$$P(\mathbf{x} \,|\, \mathbf{y}) = \frac{P(\mathbf{y} \,|\, \mathbf{x})P(\mathbf{x})}{P(\mathbf{y})} \propto P(\mathbf{y} \,|\, \mathbf{x})P(\mathbf{x}) \tag{2.3}$$

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} P(\mathbf{y} \,|\, \mathbf{x})P(\mathbf{x}) \tag{2.4}$$

$$= \arg\min_{\mathbf{x}} -\ln P(\mathbf{y} \,|\, \mathbf{x}) - \ln P(\mathbf{x}), \tag{2.5}$$

where $P(\mathbf{y} \,|\, \mathbf{x})$ is the *likelihood* or *fidelity* term, and $P(\mathbf{x})$ is the *prior* term.

The likelihood term can be derived from the image formation model in (2.1). For example, for zero-mean AWGN noise, *i.e.*, $\mathbf{n} \sim \mathcal{N}(0, \sigma^2)$, likelihood is an ℓ_2-norm:

$$P(\mathbf{y} \,|\, \mathbf{x}) \propto \exp(-\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2). \tag{2.6}$$

If the noise follows a Laplacian distribution, then the likelihood is an ℓ_1-norm:

$$P(\mathbf{y}|\mathbf{H}\mathbf{x}) \propto \exp(-\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_1). \tag{2.7}$$

For the prior term, multiple choices are available depending on target signal characteristics. For example, *total variation* (TV) regularization [91, 97]

is a popular choice in image denoising for its properties of preserving edges on the filtered image signal. For a length-N signal $\mathbf{y} = [y_1, y_2, \cdots, y_N]$, its total variation is defined as

$$V(\mathbf{y}) = \sum_{i=1}^{N-1} |y_{i+1} - y_i|. \tag{2.8}$$

TV is based on the assumption that the original signal may have large but sparse variations. Therefore, by minimizing TV of the signal—a convexification of the ℓ_0-norm [91, 16]—the reconstructed signal will be close to the original signal.

Other signal priors like *low rank prior* [27, 68], *sparsity prior* [1, 33], and *graph smoothness priors* like *graph total variation* (GTV) regularization [3, 108], *graph Laplacian regularization* (GLR) [115, 84] are also commonly used in image restoration tasks. Graph smoothness priors will be discussed in Sec. 2.3.1.

2.2 Graph Signal Processing

Graph signal processing (GSP) is an emerging field that studies signals residing on irregular data kernels described by graphs [83, 95, 21]. In this section, some basic concepts of GSP, such as graph definition, graph spectrum and graph construction, are reviewed to facilitate understanding of the subsequent chapters.

2.2.1 Graph Definition

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ consists of a node set $\mathcal{V}$ of size N and an edge set $\mathcal{E}$ specified by (i, j, w_{ij}), where $i, j \in \mathcal{V}$ and $w_{ij} \in \mathbb{R}$ is an edge weight. A graph can be *unweighted* where all edge weights have the same value, *i.e.*, $w_{ij} = 1, \forall (i, j) \in \mathcal{E}$. If a *positive graph* is *weighted*, then $w_{ij} > 0, \forall (i, j) \in \mathcal{E}$, and w_{ij} reflects the similarity between samples at nodes i and j. An *adjacency matrix* $\mathbf{W} \in \mathbb{R}^{N \times N}$ can then be defined, where $W_{ij} = w_{ij}$ if $(i, j) \in \mathcal{E}$, and $W_{ij} = 0$ otherwise. A *signed graph* with negative edge weights $w_{ij} < 0$ reflecting pairwise dissimilarity is also possible and has been studied in the literature [22]. However, in this thesis only positive graphs are used.

Graphs can be *directed* or *undirected* depending on applications. If a graph is undirected, then an edge (i, j) means node i can traverse to node j and vice versa, and $w_{ij} = w_{ji}$. This implies that $\mathbf{W}$ is a symmetric matrix. If a graph is directed, then (i, j) implies only that node i can traverse to node j. w_{ij} may not equal to w_{ji}, and $\mathbf{W}$ may not be symmetric. In some applications, *e.g.*, social networks, a user may follow another, but not the other way around, and thus a directed graph is more suitable in representing relationships in the network. In image processing, an undirected graph is more commonly used [21, 4, 115, 108, 55, 84].

A diagonal *degree matrix* is defined as $\mathbf{D} \in \mathbb{R}^{N \times N}$, where $\mathbf{D}_{ii} = \sum_j w_{ij}$. Given $\mathbf{W}$ and $\mathbf{D}$, a *combinatorial graph Laplacian matrix* is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}. \tag{2.9}$$

Since all the edge weights w_{ij} are assumed to be positive, as done in [21], one can prove that $\mathbf{L}$ is a *positive semidefinite* (PSD) matrix via the *Gershgorin*

circle theorem [105]. (Positive semidefiniteness means that $\mathbf{x}^\top \mathbf{L}\mathbf{x} \geq 0, \forall \mathbf{x}$, and the eigenvalues of $\mathbf{L}$ are non-negative, *i.e.*, $\lambda \geq 0$.)

2.2.2 Graph Spectrum

The Laplacian matrix $\mathbf{L}$ also enables a generalization of the notions of frequencies and Fourier transform to the graph signal domain. Since $\mathbf{L}$ is a real and symmetric matrix, one can show via the spectral theorem [24] that $\mathbf{L}$ is *diagonalizable*. This means that $\mathbf{L}$ has a complete set of orthonormal eigenvectors associated with real eigenvalues; *i.e.*, $\mathbf{L}$ can be eigen-decomposed into

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top, \tag{2.10}$$

where $\mathbf{U}$ is the eigen-matrix that contains eigenvectors of $\mathbf{L}$ as columns, and $\mathbf{\Lambda}$ is a diagonal matrix that contains the eigenvalues along its diagonal:

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_0 & & & \\ & \lambda_1 & & \\ & & \ddots & \\ & & & \lambda_{N-1} \end{bmatrix}. \tag{2.11}$$

Conventionally, the eigenvalues are sorted in a non-decreasing order: $0 = \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{N-1}$. $\boldsymbol{\alpha} = \mathbf{U}^\top \mathbf{x}$ is called *Graph Fourier Transform* (GFT) of $\mathbf{x}$, where $\boldsymbol{\alpha}$ is a vector of GFT coefficients, and each pair of eigenvector and eigenvalue is called a *Fourier mode* and *graph Fourier frequency*. Given a signal $\mathbf{x} = [x_1 \dots x_N]^\top$, one can filter the signal in the frequency domain by scaling

its eigenvalues, *i.e.*,

$$\mathbf{y} = \mathbf{U} \begin{bmatrix} h(\lambda_0) & & & \\ & h(\lambda_1) & & \\ & & \ddots & \\ & & & h(\lambda_{N-1}) \end{bmatrix} \mathbf{U}^\top \mathbf{x}, \tag{2.12}$$

where $h(\lambda_n)$ is a scaling factor for graph frequency λ.

There exist two normalized variants of $\mathbf{L}$: *normalized graph Laplacian*

$$\mathcal{L} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}, \tag{2.13}$$

and *random walk Laplacian*

$$\mathbf{L_{RW}} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{W}, \tag{2.14}$$

which have different eigenvalues and eigenvectors properties compared to combinatorial graph Laplacian $\mathbf{L}$. Specifically, both $\mathbf{L}$ and $\mathbf{L_{RW}}$ have a right eigenvector that is constant:

$$\mathbf{L}\mathbf{1} = \mathbf{0} = 0 \cdot \mathbf{1}, \tag{2.15}$$

$$\mathbf{L_{RW}}\mathbf{1} = \mathbf{D}^{-1}\mathbf{L}\mathbf{1} = \mathbf{0} = 0 \cdot \mathbf{1}, \tag{2.16}$$

while $\mathcal{L}$ does not. The range of the eigenvalues of $\mathcal{L}$ and $\mathbf{L_{RW}}$ is normalized to $[0, 2]$, while the eigenvalues of $\mathbf{L}$ is lower-bounded by 0. $\mathbf{L}$ and $\mathcal{L}$ are symmetric matrices, while $\mathbf{L_{RW}}$ is not. Using normalized Laplacian $\mathcal{L}$ means that the energy of filtered signal $p(\mathcal{L})\mathbf{x} = \left(\sum_{i=1}^{P} a_i \mathcal{L}^i\right)\mathbf{x}$ using polynomial filter $p(\mathcal{L}) = \sum_{i=1}^{P} a_i \mathcal{L}^i$ remains bounded. For this reason, $\mathcal{L}$ has been used in critically sampled biorthogonal graph wavelet design like GraphBio [98]. Similarly, normalized $\mathbf{L_{RW}}$ was used for patch-based JPEG image decoding [70], where it was shown that the weight of the regularization term called

(a) (b) (c)

Figure 2.1: Illustration of graph construction on images. (a) 4 connected graph; (b) a 16×16 image block, and (c) an example of graph superimposed onto it, where light lines represent smaller edge weights, and dark ones represent larger edge weights.

LeRAG computed from $\mathbf{L_{RW}}$ is easier to tune than the counterpart computed from un-normalized $\mathbf{L}$. In this thesis, for simplicity only combinatorial graph Laplacian $\mathbf{L}$ is considered, and the normalized variants $\mathcal{L}$ and $\mathbf{L_{RW}}$ are left for future work.

2.2.3 Graph Construction

In many practical scenarios in GSP, one can construct an appropriate similarity graph based on available per-node information, which can be used subsequently for graph-based processing. For example, in a temperature sensing system, each sensor node is physically placed at a planned location in a forest, and its 2D coordinate is known. In this case, one can first compute the Euclidean distance between two sensors, then compute an edge weight based on the distance [30]. In an image processing scenario, each pixel in a target image patch can be

represented by a graph node, and an edge weight can encode the estimated similarity or correlation between two neighboring pixels [21].

Specifically, similarity between nodes i and j can be coded as edge weight w_{ij}, computed based on a notion of distance $\text{dist}(i,j)$ and a Gaussian kernel, *i.e.*,

$$w_{ij} = \exp\left(-\frac{\text{dist}^2(i,j)}{\sigma^2}\right), \tag{2.17}$$

where σ is a parameter. Using a Gaussian function to define edge weight w_{ij} in (2.17) means that $0 \leq w_{ij} \leq 1$. There are many possible definitions of distance depending on applications, for example:

- *Geometric distance*: $d_l(\mathbf{l}_i, \mathbf{l}_j) = \|\mathbf{l}_i - \mathbf{l}_j\|$, where $\mathbf{l}_i \in \mathbb{R}^2$ is the 2D coordinate of the i-th pixel in a target image patch [77].
- *Photometric distance*: $d_p(x_i, x_j) = |x_i - x_j|$, where $x_i \in \mathbb{R}^+$ is the pixel intensity of the i-th pixel in a target image patch [77].
- *Saliency distance*: $d_s(s_i, s_j)$, where $s_i \in \mathbb{R}^+$ is saliency value of the i-th pixel in a target image patch [48].
- These combinations.

In recent literature, such as DeepGLR [115] and DeepGTV [108], $\text{dist}(i,j)$ was defined as *feature distance*, where the appropriate features are not manually chosen, but are learned from training data by using *convolutional neural networks* (CNN) [65, 66, 45].

The combination of geometric distance and photometric distance for computing weights is also known as *bilateral filter weights* in image processing

literature [101, 41, 85]:

$$w_{ij} = \exp\left(-\frac{\|\mathbf{l}_i - \mathbf{l}_j\|_2^2}{\sigma_l^2}\right) \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\sigma_x^2}\right), \tag{2.18}$$

where σ_l and σ_x are the two parameters. In many GSP application in image processing, a simple photometric distance is used for computing weights, *i.e.*

$$w_{ij} = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\sigma^2}\right). \tag{2.19}$$

If two pixels have a large difference in pixel intensity, like pixels on either side of a shape object boundary, then the corresponding edge weight will be small.

With an appropriately chosen underlying graph that captures inter-pixel similarities, good performance can be achieved in various imaging problems such as compression and denoising [84, 56]. An example of graph construction for an image patch is shown in Fig. 2.1c, where edge weights are computed using Eq. (2.19). The light lines between two pixels represent smaller edge weights, and the dark ones represent larger edge weights. In image processing, a 4-connected graph or 8-connected graph are commonly used towards a sparse graph construction (resulting in cheaper computation costs). If a 4-connected graph is used, as shown in Fig. 2.1c, then each pixel at a center position will be connected with its top, bottom, left, and right adjacency pixels. For an 8-connected graph, the pixels at the diagonal positions will also be connected.

2.3 Graph-based Image Processing Techniques

Two popular graph smoothness priors in the GSP literature—*Graph Laplacian Regularization* (GLR) [84, 56] and *Graph Total Variation* (GTV) [108, 3] are

first reviewed in Sec. 2.3.1. Then, some graph-based image restoration tasks, including image denoising [84], image super-resolution [57], image deblurring [3] are discussed in Sec. 2.3.2.

2.3.1 Graph Signal Priors

Figure 2.2: An example from [4] that compares GLR and GTV priors in image deblurring problem. (a) Image blurred by Gaussian blur. (b) Image filtered with SDGLR. (c) Image filtered with RGTV.

As discussed in Sec. 2.1.2, a signal prior $P(\mathbf{x})$ assumes a target signal characteristic without data—described in mathematical terms—for a MAP-formulated problem. A signal prior should be accurate in describing signal $\mathbf{x}$ *and* amenable to efficient optimization.

Graph Laplacian Regularization (GLR). By assigning sample value $x_i \in \mathbb{R}$ to node i, the ensemble $\mathbf{x} = [x_1 \dots x_N]^\top \in \mathbb{R}^N$ is called a *graph signal* on graph $\mathcal{G}$. For a given positive graph $\mathcal{G}$, signal $\mathbf{x}$ is considered smooth with respect to (w.r.t.) $\mathcal{G}$ if each sample x_i of node i is similar to samples x_j of neighboring nodes j with large edge weights w_{ij}. Equivalently in the graph

frequency domain, $\mathbf{x}$ is composed of mostly low graph frequencies, and most coefficients $\boldsymbol{\alpha} = \mathbf{U}^\top \mathbf{x}$ of high frequencies are roughly zeros. Mathematically, one can express this signal smoothness w.r.t. $\mathcal{G}$ using the *graph Laplacian regularizer* (GLR) [95]:

$$\mathbf{x}^\top \mathbf{L} \mathbf{x} = \sum_{(i,j) \in \mathcal{E}} w_{ij} (x_i - x_j)^2 = \sum_{k=1}^{N} \lambda_k \alpha_k^2, \tag{2.20}$$

where $\mathbf{L}$ is the graph Laplacian matrix, α_k is the k-th GFT coefficient, and λ_k is the k-th frequency (eigenvalue of $\mathbf{L}$). A small GLR thus implies signal $\mathbf{x}$ is smooth w.r.t. graph $\mathcal{G}$.

Note that in this GLR definition, edge weights w_{ij} are fixed after initialization. In contrast, in some recent literature [70, 84] *signal-dependent* GLR (SDGLR) is proposed. Specifically, the weight in SDGLR is computed with the signal, and hence the Laplacian matrix is a function of the signal $\mathbf{x}$, *i.e.*, $\mathbf{x}^\top \mathbf{L}(\mathbf{x}) \mathbf{x}$. As an example, one can compute the edge weight in SDGLR using Eq. (2.19), which computes an edge weight using corresponding pixel values.

SDGLR tends to perform better than GLR in image restoration because SDGLR promotes *piecewise smoothness* (PWS) in reconstructed signals. Specifically, minimizing GLR only promotes $(x_i - x_j)^2$ to 0 for large edge weights w_{ij}. In contrast, to minimize each term in the sum in Eq. (2.20) for SDGLR, either $(x_i - x_j)^2$ becomes close to 0, or $(x_i - x_j)^2$ becomes very large, in which case w_{ij} becomes close to 0. This results in a bimodal edge weight distribution, and a PWS reconstructed signal. PWS is commonly used as a prior for image restoration which assumes sharp object boundaries and slowly varying interior surfaces.

Graph Total Variation (GTV). Another popular graph smoothness

prior is *Graph Total Variation* (GTV), which can be defined as

$$\|\mathbf{x}\|_{\text{GTV}} = \sum_{(i,j)\in\mathcal{E}} w_{ij}|x_i - x_j|, \tag{2.21}$$

In this case, w_{ij} is fixed after initialization. Unlike GLR, GTV does not have a natural frequency interpretation.

In [3], GTV is extended to *reweighted graph total variation* (RGTV),

$$\|\mathbf{x}\|_{\text{RGTV}} \sum_{(i,j)\in\mathcal{E}} w_{ij}(x_i, x_j)|x_i - x_j|, \tag{2.22}$$

where each edge weight w_{ij} is a function of samples x_i and x_j in $\mathbf{x}$ and thus is *signal-dependent*. In particular, one can use Eq. (2.19) to compute w_{ij}. Compared with GTV which promotes $d = |x_i - x_j|$ to 0, to minimize (2.22) RGTV promotes d to either a very small value (*i.e.*, $|x_i - x_j|$ becomes small) or very large (*i.e.*, $w_{ij}(x_i, x_j)$ becomes very small). Thus, RGTV promotes a bimodal distribution of edge weights, and the resulting image patch becomes PWS as well.

Example images reconstructed using SDGLR and RGTV as image priors in an image deblurring problem are shown in Fig. 2.2. In Fig. 2.2(c), one can see that the edges on the filtered image using RGTV can be better preserved compared with the one using SDGLR. An analysis in [4] shows that RGTV and SDGLR can both promote the desirable bimodal edge weight distribution, while the signal independent ones cannot. However, the promotion of bimodal edge weight distribution using SDGLR tends to slow down significantly when $d = |x_i - x_j|$ approaches 0, while this problem can be overcome using RGTV.

2.3.2 GSP in Image Restoration

GSP tools have been successfully applied in many image restoration problems. Some related works using GLR and RGTV in MAP formulations for restoration tasks are discussed below.

Image denoising. With the assumption of AWGN noise, one can write the image denoising problem via MAP using GLR as signal prior as

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \|\mathbf{x} - \mathbf{y}\|_2^2 + \mu \mathbf{x}^\top \mathbf{L}\mathbf{x}, \tag{2.23}$$

where $\mu > 0$ is a tradeoff parameter. This formulation has a closed-form solution given a fixed $\mathbf{L}$:

$$\mathbf{x}^* = (\mathbf{I} + \mu\mathbf{L})^{-1}\mathbf{y}. \tag{2.24}$$

In [84], the authors derive the optimal GLR (OGLR) for the image denoising task by performing analysis for a signal model in the continuous domain.

Image super-resolution. In [57], the authors pose a super-resolution problem as a joint problem with image denoising. GLR is used to promote piecewise smoothness for depth images in this work:

$$\mathbf{x}^* = \|\mathbf{D}\mathbf{H}\mathbf{x} - \mathbf{y}\|_2^2 + \lambda \mathbf{x}^\top \mathbf{L}\mathbf{x}, \tag{2.25}$$

where $\mathbf{D}$ is the downsample matrix, and $\mathbf{H}$ is the Gaussian low-pass filtering operation. Like denoising, this problem also has a closed-form solution given fixed $\mathbf{D}$, $\mathbf{H}$ and $\mathbf{L}$.

Image deblurring. In [3], RGTV is used as a signal prior to formulate a MAP problem for blind image deblurring. Given an observed blurred image

patch $\mathbf{b}$, the problem is formulated as

$$(\mathbf{x}^*, \mathbf{k}^*) = \arg\min_{\mathbf{x},\mathbf{k}} \frac{1}{2}\|\mathbf{x} \otimes \mathbf{k} - \mathbf{b}\|_2^2 + \lambda \|\mathbf{x}\|_{\text{RGTV}} + \mu \|\mathbf{k}\|_2^2, \tag{2.26}$$

where $\otimes$ is a convolution operator, $\mathbf{k}$ is the unknown blur kernel, and $\lambda > 0$ and $\mu > 0$ are parameters. For unknown blur kernel $\mathbf{k}$, as conventionally done in the blind deblurring literature [2, 76, 15], an ℓ_2-norm $\|\mathbf{k}\|_2^2$ is used as a prior for $\mathbf{k}$. Solving for both signal $\mathbf{x}$ and kernel $\mathbf{k}$ simultaneously is difficult. Hence, (2.26) is solved alternately in [3], where $\mathbf{x}$ is fixed while an optimal $\mathbf{k}$ is computed and vice versa until convergence.

Other graph-based image restoration tasks. In [70], as an alternative to GLR, a graph regularization term called LeRAG defined using the random walk graph Laplacian matrix instead of the combinatorial Laplacian matrix is proposed for the problem of soft decoding of JPEG images. The advantage of LeRAG is that it is defined using a normalized Laplacian matrix, while the first (right) eigenvector is still a constant vector, which is desirable for smooth images. In [55], a unified framework that alternately optimizes the graph (image structure) and the signal on top of the graph (pixel patch) is proposed. This is done to promote PWS in edge weights (interpreted as a graph signal in a dual graph), which is a reasonable assumption for piecewise constant (PWC) images like depth maps. In [19], a new *gradient graph Laplacian regularizer* (GGLR) is proposed to promote piecewise planar (PWP) signal reconstruction for the image interpolation problem.

This work leverages these earlier works in graph spectral image restoration, but differs in that, in this work, (noisy) pixel observations are not directly observable. Instead, an appropriate graph needs to be constructed using multiple observed color pixels in the Bayer-patterned grid to estimate edge

weights by exploiting inter-color correlation.

2.4 Joint Demosaicking Problems

As discussed in Chapter 1, since a camera sensor only captures one color channel at each pixel location in a Bayer-patterned grid, demosaicking is conventionally performed to interpolate missing color pixels early in the image processing pipeline. Demosaicking is a well-studied image processing problem [47, 117, 75, 118, 44]. Instead of performing demosaicking in isolation, in recent years it is often studied in tandem with other image processing tasks for more optimal end-to-end performance. An overview of joint demosaicking problems is provided in this section.

2.4.1 Joint Demosaicking and Denoising

One of the most widely studied joint demosaicking problems is joint demosaicking / denoising (JDD) [52, 116, 42, 28]. Given a noise-corrupted Bayer-patterned image, the joint demosaicking / denoising problem is to directly estimate the original image with full color channels. A separate approach that performs demosaicking followed by denoising would spread the original additive noise to neighboring pixel locations via interpolation during the demosaicking step, introducing correlated noise, and making the subsequent denoising problem more challenging [80, 119].

To address the above problem, JDD problem is investigated to perform denoising and demoaicking at the same time. In [53] a joint demosaicking /

denoising approach is proposed based on a total least squares image denoising method. In [62], the JDD problem is solved by learning a sequence of energy minimization problems, and a proposed algorithm can be applied for different sensor layouts such as Fujifilm X-Trans pattern. A machine learning based approach is proposed in [59], which performs JDD through a learned nonparametric random field.

In some recent literature, deep-learning-based approaches are used for the JDD problem, which is reported to outperform the model-based approaches discussed above. A deep-learning based JDD method is first proposed in [43]. In this work, the authors observe that simply using large quantity of training samples do not guarantee convincing JDD performance, and thus they propose a dataset that include some challenging patches which is useful in training a network for the JDD problem. In [69], a network is proposed, which interpolates pixels of green channel first as a guidance to recover the missing values of the other color channels. A JDD method based on generative adversarial network (GAN) is proposed in [29]. Although these learning-based approaches can perform well if the testing image shares a similar distribution with the training data, it can perform poorly if the noise statistics differ from the training data [31].

This thesis shares a similar optimization philosophy as JDD, but focus on designing a fast optimization algorithm for the joint demosaicking / rectification problem. Moreover, compared to JDD, this problem is more complicated in that pixels in the rectified image grid are typically mapped to non-integer locations in the Bayer-patterned image grid.

2.4.2 Joint Demosaicking and Rectification

The joint demosaicking / rectification problem is relatively new compared to JDD. Existing works on this problem [61, 58, 63] focus on efficient implementation of a simple interpolation scheme for the joint demosaicking / rectification process on hardware, *e.g.*, implementing a linear interpolation algorithm with circuits to interpolate the target pixels at non-integer locations, such that the algorithm can be embedded on chips.

Recently, a joint demosaicking / rectification algorithm is proposed in [106]. This algorithm is implemented on an *field-programmable gate array* (FPGA) hardware system. In this work, the authors consider that embedded camera systems often are limited by their throughput capabilities due to hardware resource limitations. Thus, they propose a demosaicking and rectification algorithm to leverage its efficiency to reduce the memory footprint and improve streaming performance.

2.4.3 Other Joint Demosaicking and Image Restoration Problems

A brief review of other joint demosaicking problems is also provided. However, these problems are quite different from the joint demosaicking / rectification problem, and thus a new solution is proposed in this thesis.

Joint Demosaicking and Deblurring. In [23], a multi-scale deep convolutional neural network is used to solve demosaicking and blind deblurring jointly. In [73], a joint demosaicking and deconvolution algorithm based on

first-order primal-dual minimization is proposed.

Joint Demosaicking and Super Resolution. In [35], a multi-frame joint demosaicking and super resolution algorithm is proposed based on a MAP estimation technique. A joint demosaicking and super-resolution algorithm for unregistered aliased images is proposed in [104], which computes the alignment parameters between the images on the raw camera data first before performing interpolation. Some deep-learning based approaches for this problem are proposed in [112, 121, 51].

Chapter 3

Problem Formulation

In this chapter, a reverse mapping from Bayer-pattern grid to a rectified grid is first discussed in Sec. 3.1. In Sec. 3.2, a Maximum a Posterior (MAP) formulation for joint demosaicking and rectification problem is proposed. In Sec. 3.3, a graph construction method for the rectified image is proposed, where the edge weights are computed with estimated gradients. The inter-pixel gradient estimation method is discussed in Sec. 3.4.

3.1 Reverse Mapping of Image Grids

From OpenCV [37], a mapping function, $f : \mathbb{Z}^2 \rightarrow \mathbb{R}^2$, is obtained that maps a pixel i's integer 2D coordinate (x_i, y_i), $x_i, y_i \in \mathbb{Z}$, in the rectified image grid, to a real 2D coordinate (x_s, y_s), $x_s, y_s \in \mathbb{R}$, in the fisheye camera Bayer-patterned image grid. Specifically, this function is derived from estimated

camera distortion parameters and targeted output camera intrinsic parameters[5], which will be discussed in detail in Sec. 4 that discusses data creation. See Fig. 3.1 for an illustration of reverse pixel mapping from two adjacent pixels i and j in the rectified image grid to locations s and t in the Bayer-patterned grid.

In a conventional demosaicking algorithm [26], each color pixel (say red) in an image grid is interpolated as a weighted linear combination of neighboring red pixels. Similarly, an N-pixel color block $\mathbf{x} \in \mathbb{R}^N$ in the rectified image grid can be linearly interpolated from an M-pixel neighborhood $\mathbf{y} \in \mathbb{R}^M$ in the Bayer-patterned grid as $\mathbf{Hy}$, where $\mathbf{H} \in \mathbb{R}^{N \times M}$ is a weight matrix used for interpolation. More complex non-linear interpolation methods [11, 49] such as NLM [118] can also be incorporated into the joint demosaicking / rectification framework—this direction is left for future work.

3.2 MAP Formulations for Image Restoration

The goal is to reconstruct a target square pixel patch $\mathbf{x}$ in the rectified grid in a chosen color component, given an M-pixel neighborhood patch $\mathbf{y}$ of the same color in the Bayer-patterned grid. Using GLR [84], an optimization problem can be formulated via MAP as

$$\min_{\mathbf{x}} \|\mathbf{Hy} - \mathbf{x}\|_2^2 + \mu\, \mathbf{x}^\top \mathbf{L}_\mathbf{x} \mathbf{x} \tag{3.1}$$

where $\mu > 0$ is a weight parameter to trade off the (first) fidelity term with the (second) signal prior. In words, (3.1) states that the reconstructed signal $\mathbf{x}$

[5]More information about this mapping function can be found in: `https://docs.opencv.org/master/db/d58/group__calib3d__fisheye.html`

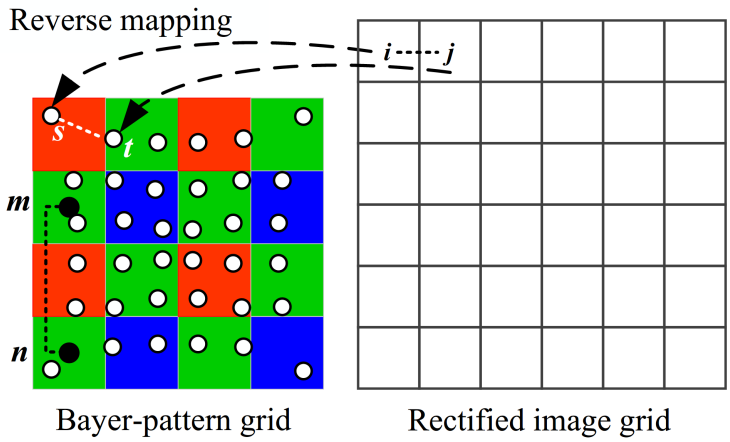


Figure 3.1: Reverse mapping from the rectified grid to the Bayer-patterned grid. The white circles at the non-integer locations are the mapped locations from the rectified grid. The black circles at the integer locations represent captured pixels in the same color channel on the Bayer-patterned grid. Pixel intensity differences at these locations are used to estimate edge weights between the white circles.

should be similar to interpolation $\mathbf{Hy}$ while being smooth with respect to a graph specified by graph Laplacian matrix $\mathbf{L_x}$.

Note that, although the graph smoothness prior is introduced in (3.1) to combat possible inaccuracy in interpolation $\mathbf{Hy}$ during the demosaicking process, (3.1) is not explicitly formulated as an image denoising problem. Instead, it is formulated as an *image restoration* problem, and the goal is to reconstruct the target signal $\mathbf{x}$ from observation $\mathbf{y}$ during the restoration process.

For fixed interpolation matrix $\mathbf{H}$ and graph Laplacian $\mathbf{L_x}$, (3.1) is an unconstrained *quadratic programming* (QP) problem, whose solution can be

computed from a system of linear equations:

$$(\mathbf{I} + \mu \mathbf{L}_{\mathbf{x}})\,\mathbf{x} = \mathbf{H}\mathbf{y}. \tag{3.2}$$

Since the coefficient matrix $\mathbf{I} + \mu\,\mathbf{L}_{\mathbf{x}}$ in (3.2) is sparse, symmetric and positive definite (PD), (3.2) can be efficiently solved without matrix inverse using a fast numerical linear algebra algorithm such as *conjugate gradient* (CG) [79]. One can prove its positive definiteness as follows.

Proof: The positive definiteness of $\mathbf{I}$ and positive semidefiniteness of $\mathbf{L}_{\mathbf{x}}$ implies that $\mathbf{x}^\top\mathbf{I}\mathbf{x} = \mathbf{x}^\top\mathbf{x} > 0, \forall \mathbf{x} \neq \mathbf{0}$, and $\mathbf{x}^\top\mathbf{L}_{\mathbf{x}}\mathbf{x} \geq 0, \forall \mathbf{x}$. Thus, for any signal $\mathbf{x} \neq \mathbf{0}$,

$$\mathbf{x}^\top(\mathbf{I} + \mu\mathbf{L}_{\mathbf{x}})\mathbf{x} = \mathbf{x}^\top\mathbf{I}\mathbf{x} + \mathbf{x}^\top\mathbf{L}_{\mathbf{x}}\mathbf{x} \tag{3.3}$$

$$\geq \mathbf{x}^\top\mathbf{I}\mathbf{x} > 0. \tag{3.4}$$

Thus, $\mathbf{I} + \mu\,\mathbf{L}_{\mathbf{x}}$ is a PD matrix.

3.3 Graph Construction

The restoration performance of (3.2) depends heavily on how the underlying graph is constructed for target patch $\mathbf{x}$ in the rectified 2D grid, which determines Laplacian $\mathbf{L}_{\mathbf{x}}$. For connectivity of patch $\mathbf{x}$, an 8-connected graph is used in this work, where each pixel in $\mathbf{x}$ is connected to its immediate vertical, horizontal and diagonal neighbors. For edge weight w_{ij} that connects pixel pair (i, j) in $\mathbf{x}$, conventionally it is inversely proportional to the *feature distance* of the two corresponding nodes; *i.e.*, the larger the feature distance, the smaller the edge weight [96]. In this imaging scenario, the feature distance is assumed to be the

magnitude of the estimated *signal gradient* $\Delta_{ij} \in \mathbb{R}^+$ between samples i and j. Using an exponential function as the kernel, the weights w_{ij} can be written as

$$w_{ij} = \exp\left(-\frac{\Delta_{ij}^2}{\sigma_w^2}\right) \tag{3.5}$$

where σ_w is a parameter. (3.5) implies that $0 \leq w_{ij} \leq 1$, and the resulting graph Laplacian $\mathbf{L_x}$, as defined in Section 2.3.1, is positive semi-definite (PSD) (see [20] for a proof using Gershgorin Circle Theorem).

The crux of the graph construction procedure thus rests in the gradient estimation for a pixel pair (i, j). The gradient $\Delta_{ij} \in \mathbb{R}^+$ is estimated via a *maximum likelihood estimation* (MLE) formulation as follows. Suppose there are K noisy observations δ_{ij}^k of Δ_{ij}, $k \in \{1, \ldots, K\}$, available. Then MLE of Δ_{ij} given δ_{ij}^k is:

$$\max_{\Delta_{ij}} \Pr\left(\Delta_{ij} \mid \{\delta_{ij}^k\}_{k=1}^K\right) \rightarrow \max_{\Delta_{ij}} \prod_{k=1}^{K} \Pr\left(\delta_{ij}^k \mid \Delta_{ij}\right) \tag{3.6}$$

where in (3.6) each noisy observation δ_{ij}^k is assumed to be generated independently, each with the following distribution:

$$\Pr\left(\delta_{ij}^k | \Delta_{ij}\right) \propto \exp\left\{-v_{ij}^k (\Delta_{ij} - \delta_{ij}^k)^2\right\} \tag{3.7}$$

where v_{ij}^k is a unique parameter for random variable δ_{ij}^k, to be discussed in Section 3.4 in detail.

Minimizing the negative log of likelihood (3.6) results in

$$\Delta_{ij}^* = \arg\min_{\Delta_{ij}} \sum_{k=1}^{K} v_{ij}^k (\Delta_{ij} - \delta_{ij}^k)^2. \tag{3.8}$$

To solve (3.8), one can take the derivative with respect to Δ_{ij} and set it to 0, resulting in

$$\Delta_{ij}^* = \frac{1}{V} \sum_{k=1}^{K} v_{ij}^k \delta_{ij}^k \tag{3.9}$$

where $V = \sum_{k=1}^{K} v_{ij}^k$. In other words, the optimal solution Δ_{ij}^* in (3.9) is a weighted average of the noisy gradient observations δ_{ij}^k.

3.4 Noise Model for Inter-pixel Gradient

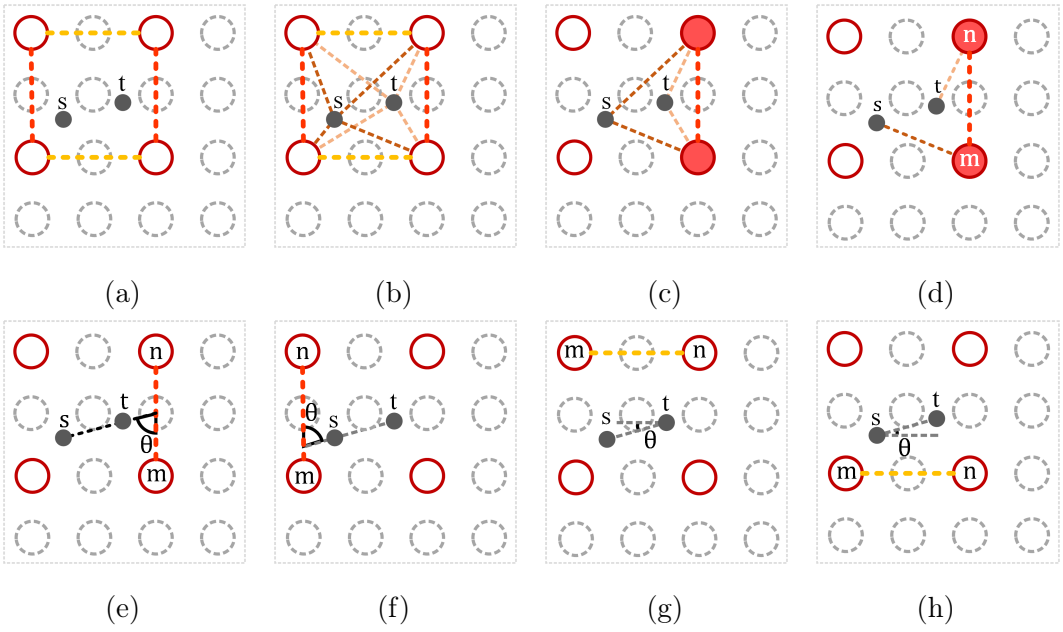


Figure 3.2: Pairing examples. (a) Discover pairs $(m, n) \in \mathcal{N}_{ij}$. (b) Compute geometric distances between s, t and each pixel in (m, n). (c) Group (s, m) and (t, n) for each pair (m, n) according to geometric distance. (d) Pairs (s, m) and (t, n) are designated together into a group. (e)-(h) Compute the angle between (m, n) and (s, t).i

One can obtain K "noisy" gradient observations δ_{ij} in this joint demosaicking / rectification framework as follows. First, define a spatial neighborhood $\mathcal{N}_{ij}$ surrounding non-integer locations s and t in the Bayer-patterned grid that correspond to pixel pair (i, j) in the rectified grid. Within neighborhood $\mathcal{N}_{ij}$,

pixel pairs $(m, n) \in \mathcal{N}_{ij}$ can be discovered that are pairs of adjacent captured pixels of the same color in the Bayer-patterned grid. For example, as shown in Fig. 3.2a, given $\mathcal{N}_{ij}$ that is a 4-by-4 image patch, there are four pixels that belong to the red channel. If every two adjacent pixels in horizontal or vertical directions are considered as pairs, then within $\mathcal{N}_{ij}$, four pairs (m, n) belonging to the red channel can be discovered, as highlighted in red and orange color lines for vertical and horizontal pairs in Fig. 3.2a. Similarly, for other color channels, pairs can be discovered and used to compute the estimated gradient for each channel.

The gradient for pair $(m, n) \in \mathcal{N}_{ij}$ is first computed as

$$\delta_{ij}^{mn} = y_m - y_n \tag{3.10}$$

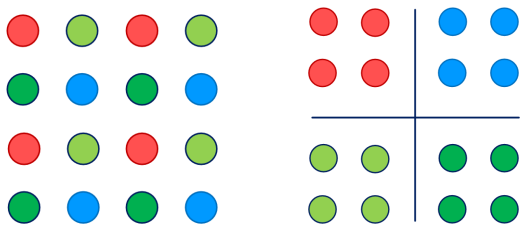
where y_m and y_n are the pixel intensity corresponding to the pixel pair (m, n) on the Bayer-patterned grid.

An associated weight v_{ij}^{mn} for pixel pair (m, n) is computed next as

$$v_{ij}^{mn} = \exp\left\{-\frac{\|\mathbf{l}_s - \mathbf{l}_m\|_2^2 \, \|\mathbf{l}_t - \mathbf{l}_n\|_2^2}{\sigma_v^2}\right\} \cos\theta_{st}^{mn} \, \rho_{st}^{mn} \tag{3.11}$$

where $\mathbf{l}_s$ is the coordinate of pixel s in the Bayer-patterned grid. As an example shown in Fig. 3.2a to 3.2d, (s, m) and (t, n) are grouped as follows.

1. In Fig. 3.2a, candidate pairs (m, n) in $\mathcal{N}_{ij}$ are first discovered.
2. In Fig. 3.2b, geometric distances between each pixel in pairs (m, n) and each pixel in (s, t) are computed.
3. From Fig. 3.2c to Fig. 3.2d for each pair (m, n), based on the geometric distances computed in the last step, designate two pairs (s, m) and (t, n)



(a) (b)

Figure 3.3: An example for computing correlation factor.

together into a *group*: label a pixel in pair (m, n) that is closer to s as m, and the other pixel as n.

Angle is used as a feature for differentiating the contribution made by the gradient in vertical and horizontal directions for computing (s, t). θ_{st}^{mn} is the acute angle between line (s, t) and line (m, n), which can be computed once the pairing between (m, n) and (s, t) is determined. As examples shown in Fig. 3.2e to 3.2h, angle θ_{st}^{mn} is computed for the 4 pairs (m, n) and (s, t).

ρ_{st}^{mn} is a *color gradient correlation factor* that estimates the local correlation of color gradients between colors of pairs (m, n) and (s, t), where pairs (m, n) and (s, t) may belong to different color channels. Specifically, when computing ρ for red and blue channels, the M-pixel Bayer-patterned patch $\mathbf{y} \in \mathbb{R}^{\sqrt{M} \times \sqrt{M}}$ is reshaped into 4 submatrices $\tilde{\mathbf{y}} \in \mathbb{R}^{\frac{\sqrt{M}}{2} \times \frac{\sqrt{M}}{2}}$ based on color channels, where within each 4-pixel RGGB array on the Bayer-pattern, pixels of green channel in the diagonal location is interpreted as two independent channels—Green-1 and Green-2. An example is shown in Fig. 3.3a-3.3b. In the case of the red channel, the 2D correlation coefficients between it and the other 3 channels (Blue, Green-1, Green-2) are computed as $\rho_{RB}, \rho_{RG_1}, \rho_{RG_2}$, and its self-correlation coefficient is $\rho_{RR} = 1$. The coefficients of the blue channel are computed

in the same way. For the green channel, the maximum value from the two computed correlation coefficients between it and the other two channels (Red and Blue), *i.e.*, $\rho_{RG} = \max\{\rho_{RG_1}, \rho_{RG_2}\}$ and $\rho_{BG} = \max\{\rho_{BG_1}, \rho_{BG_2}\}$ is used. See Algorithm 1 for a summary of the proposed algorithm.

Algorithm 1 Joint demosaicking / rectification method.

Input: Bayer-pattern image patch $\mathbf{y}$, $\mathbf{H}$.

Output: Target image patch $\mathbf{x}^*$.

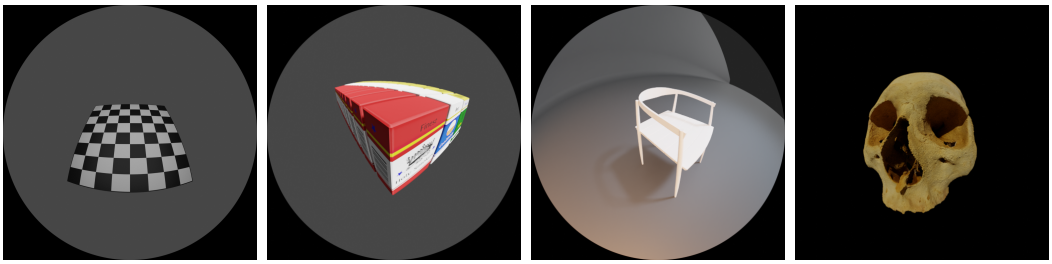
for *each pair* (i, j) *on targeted image patch* **do**
    Locating pair (s, t) on Bayer pattern $\mathbf{y}$ with $\mathbf{H}$
    Compute the correlation factor ρ with observations in $\mathcal{N}_{ij}$
    Compute the gradient $\delta_{i,j}^{m,n}$ and weight $v_{i,j}^{m,n}$ with each pair $(m, n) \in \mathcal{N}_{ij}$ in three channels with (3.10) and (3.11).
    For each channel, compute the estimated gradient $\Delta_{i,j}^*$ by weighted average via (3.9).
    For each $\Delta_{i,j}^*$, compute the edge weight $w_{i,j}$ with (3.5).
end for
    Compute the initial $\mathbf{L}_\mathbf{x}$ via $\mathbf{L}_\mathbf{x} = \mathbf{D} - \mathbf{A}$.
while *not converge* **do**
    Solving $\mathbf{x}^*$ via (3.2).
    Update $\mathbf{L}_\mathbf{x}^*$ based on $\mathbf{x}^*$ with (3.5).
end while

Chapter 4

Building Ground-truth Datasets

To objectively evaluate the proposed joint demosaicking and rectification scheme described in Chapter 3, a comprehensive dataset containing ground-truth image pairs both in the fisheye camera's Bayer-patterned grid and in the rectified image grid is necessary. Unfortunately, this kind of dataset is difficult to obtain in practice. This is because one typically cannot set up a fisheye camera *and* a regular camera at exactly the same viewing angle to capture a 3D scene at exactly the same time.

An alternative solution is to generate synthetic images for evaluation. However, in many synthetic camera image datasets, such as a dataset proposed in [32], only a synthesized fisheye camera image is provided, but not a corresponding rectified image. Thus, in this thesis, a synthetic image dataset that includes image pairs on both the fisheye camera's Bayer-patterned grid and the rectified image grid is constructed. These images are rendered using 3D computer graphics models and two camera models via an open source 3D



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

Figure 4.1: Example fisheye camera images generated with software *blender* [7]. (a) Chessboard image for estimating reverse mapping matrix. (b)-(d) Fisheye images generated with three 3D models: `Boxes`, `Chair` and `Skull`.

software called *blender* [7]. 3D objects are captured from different viewing angles, rendered as images in spatial resolution of 512×512, and saved in storage via lossless compression in BMP format to populate the dataset.

Four 3D models are used to generate the dataset, which consists of more than 100 high-quality images. A camera intrinsic parameter matrix is also derived for the fisheye camera model, which is used to compute the reverse mapping matrix. The details of this generation process are described in this chapter.

4.1 Camera Modeling

A perspective camera is a mathematical model of an ideal pinhole camera that follows perspective projection. As illustrated in Fig. 4.2, for images captured with a perspective camera without any lens distortion, its *field of view*[6] (FOV)

[6]Detailed description of FOV can be found at :`https://en.wikipedia.org/wiki/Angle_of_view`

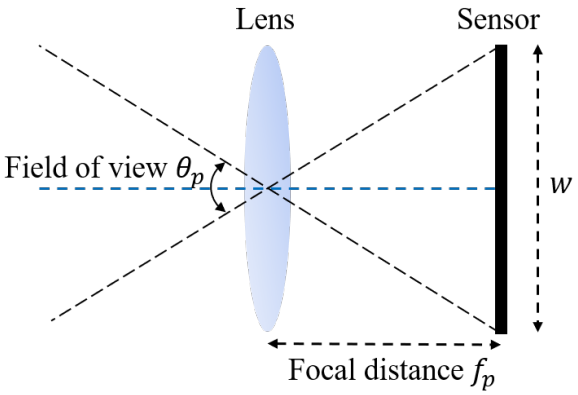


Figure 4.2: Field of view (FOV) of perspective camera.

can be modeled as

$$\theta_p = 2 \arctan\left(\frac{w}{2f_p}\right) \tag{4.1}$$

where f_p is the focal distance. w is the width of the sensor in millimeters, which in a full frame camera ($36mm \times 24mm$) is 36. For the fisheye camera, there are 4 popular models as discussed in [6]. To simplify the generation process, the simplest one, *i.e.*, equidistant model, is adopted in this thesis, which is modeled as

$$\theta_f = \frac{h}{f_f} \tag{4.2}$$

where f_f and θ_f are respectively the focal distance and FOV in this model, and h is the height of the sensor in millimeters, which is 24 for a full-frame camera. Note that there is no direct mapping function between these two models, and thus one can only estimate the corresponding parameters of the fisheye camera in the pinhole camera model, such that the reverse mapping function can be computed.

4.2 Camera Parameter Estimation

OpenCV [9] is used to estimate the parameters of fisheye camera in the perspective camera model. First, 150 checkerboard images with the fisheye camera are captured, as shown in Fig. 4.1a. They are then sent to the calibration algorithm to estimate the intrinsic parameter matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$ and distortion coefficients $\mathbf{d}$. Matrix $\mathbf{K}$ is defined as

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.3}$$

f_x and f_y are focal distances in x- and y-axis respectively, and c_x and c_y are repectively the x- and y-coordinates of image center of the fisheye lens in the perspective camera model. $\mathbf{d} \in \mathbb{R}^5$ is a vector for calibrating radical and tangential distortion. Then, the targeted camera matrix $\mathbf{P} \in \mathbb{R}^{3\times3}$ needs to be specified to generate the reverse mapping function, where

$$\mathbf{P} = \begin{bmatrix} f_{p,x} & 0 & c_{p,x} \\ 0 & f_{p,y} & c_{p,y} \\ 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

and $f_{p,x}$ and $f_{p,y}$ are the focal distances of the targeted perspective lens. Note that if a large FOV θ_f in fisheye camera model, *e.g.*, 180°, and a large focal distance $f_{p,x}$ and $f_{p,y}$ in perspective camera model (small FOV) *e.g.*, $50mm$, FOV = 39.6° are chosen, then the rectified image will be too smooth, since a rectified image is only interpolated with a small portion of pixels of the fisheye camera image. To avoid this problem, an FOV $\theta_f = 180°$ for fisheye camera and a wide-angle lens focal distance $f_{p,x} = f_{p,y} = 16mm$ for the perspective camera are chosen. Once the intrinsic parameter matrix $\mathbf{K}$ is computed, a

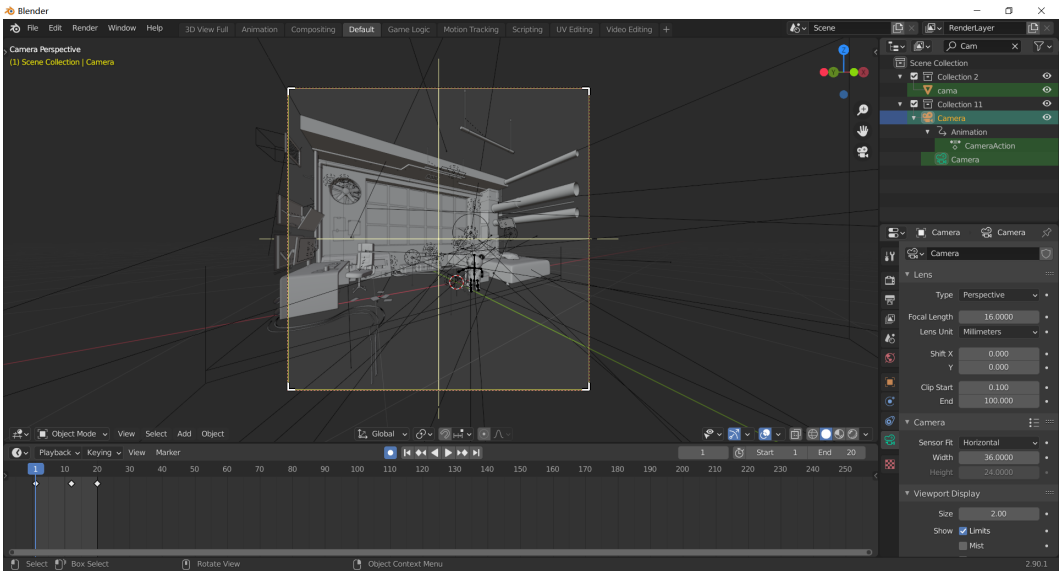


Figure 4.3: 3D Modeling in *blender*.

mapping matrix that maps each pixel location on the targeted grid to the Bayer-pattern grid is computed using OpenCV [9].

4.3 Data Generation

The user interface of 3D modeling in software *blender* is shown in Fig. 4.3. Two camera models are used to capture the object from different positions. The specification of the camera settings is shown in Fig. 4.4. For the perspective camera, the focal length is set to $16mm$, and for fisheye camera, the FOV is set to 180° and a full-frame camera sensor size is used. In some 3D models, the depth of field and aperture features are enabled to better simulate realistic camera capturing settings.

These frames are rendered with the highest render quality settings. For

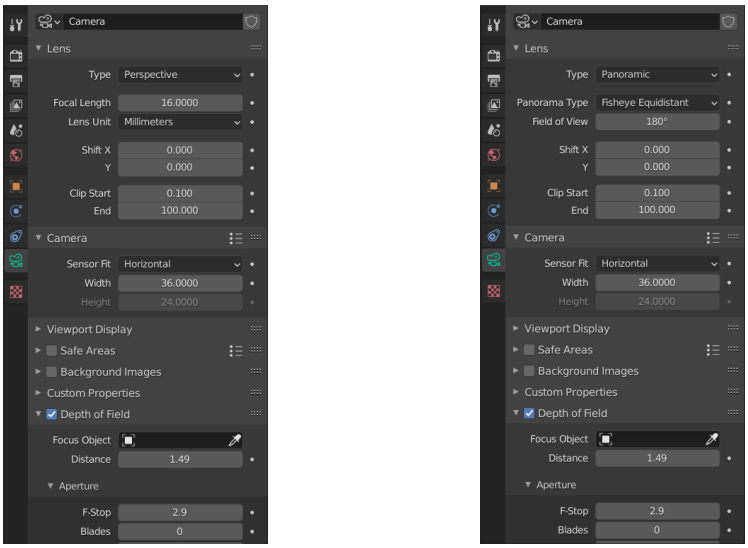


Figure 4.4: Camera settings.

example, *ray-tracing* for those 3D models that support this feature is enabled to get a more realistic simulation of environmental light settings. In addition, *increase sampling rate* is set to a high value to ensure that the rendered image is noise-free. The frames are saved with lossless compression into BMP format after the rendering procedure.

Chapter 5

Experiments

In this chapter, results of experiments to objectively evaluate algorithm performance are presented. In Sec. 5.1, an experimental setup is first described. In Sec. 5.2, quantitative comparisons between the proposed method and competing methods are discussed.

5.1 Experimental Setup

The proposed joint demosaicking / rectification algorithm was tested on a Multi-FoV image dataset [123] and the in-house constructed dataset[7]. The Multi-FoV image dataset includes two scenes: `room` and `city`. 5 images from `room` and 25 images from `city` were used in the experiment. The in-house dataset includes 140 pinhole and fisheye camera images generated from 4 publicly available

[7]The dataset is available at: https://github.com/fengbolan/York-Fisheye-Image-Rectification-Dataset

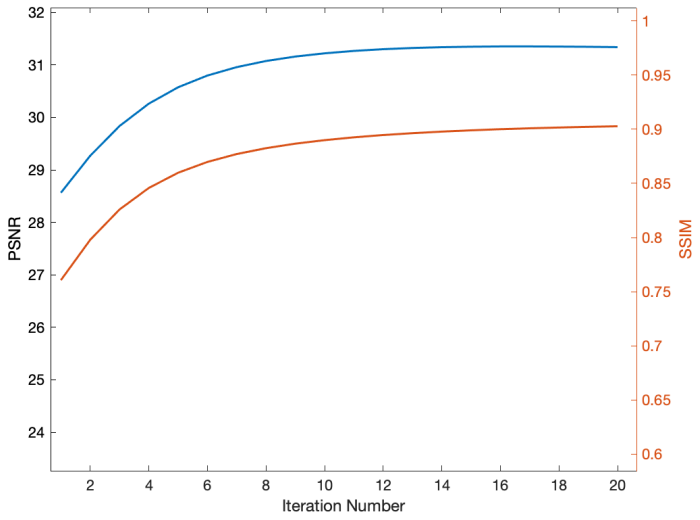
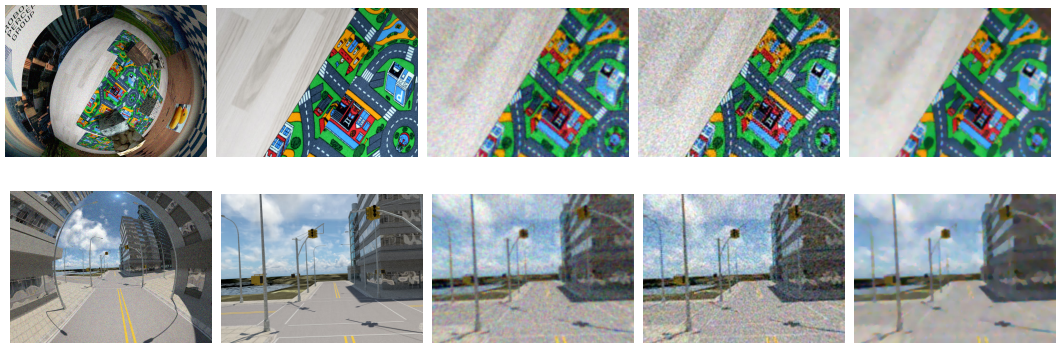


Figure 5.1: PSNR and SSIM of a rectified image under different iteration number.

3D models: `box`, `chair`, `skull` and `teddy`. 3 images from each scene were used for evaluation. For demosaicking, two competing schemes were employed: 1) bilinear interpolation, and 2) a high quality linear (HQL) filter [74]. For rectification, a bilinear interpolation method was employed. Given a fisheye image, as shown in Fig. 5.2a and 5.4a, an image region corresponding to the ground truth image (rectified image) was designated as the region of interest (ROI), shown in Fig. 5.2b and 5.4b, respectively. Then color pixels were removed in the ROI to generate a pre-demosaick Bayer-patterned image with additive Gaussian noise as the input of competing algorithms.

The parameters of the proposed algorithm were set empirically according to the content of the images. As described in Sec. 3.4, the algorithm was performed iteratively. In Fig. 5.1, relation between the restoration performance, which was evaluated with peak signal to noise ratio (PSNR) and structural index



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

Figure 5.2: Results of demosaicking and rectification for `room` and `city` [123]. (a) Ground truth fisheye camera image. (b) Ground truth pinhole image. (c) Demosaicking and rectification using the bilinear method. (d) Demosaicking using high quality linear interpolation (HQL) [74] and rectification using the bilinear method. (e) The proposed joint demosaicking / rectification method.

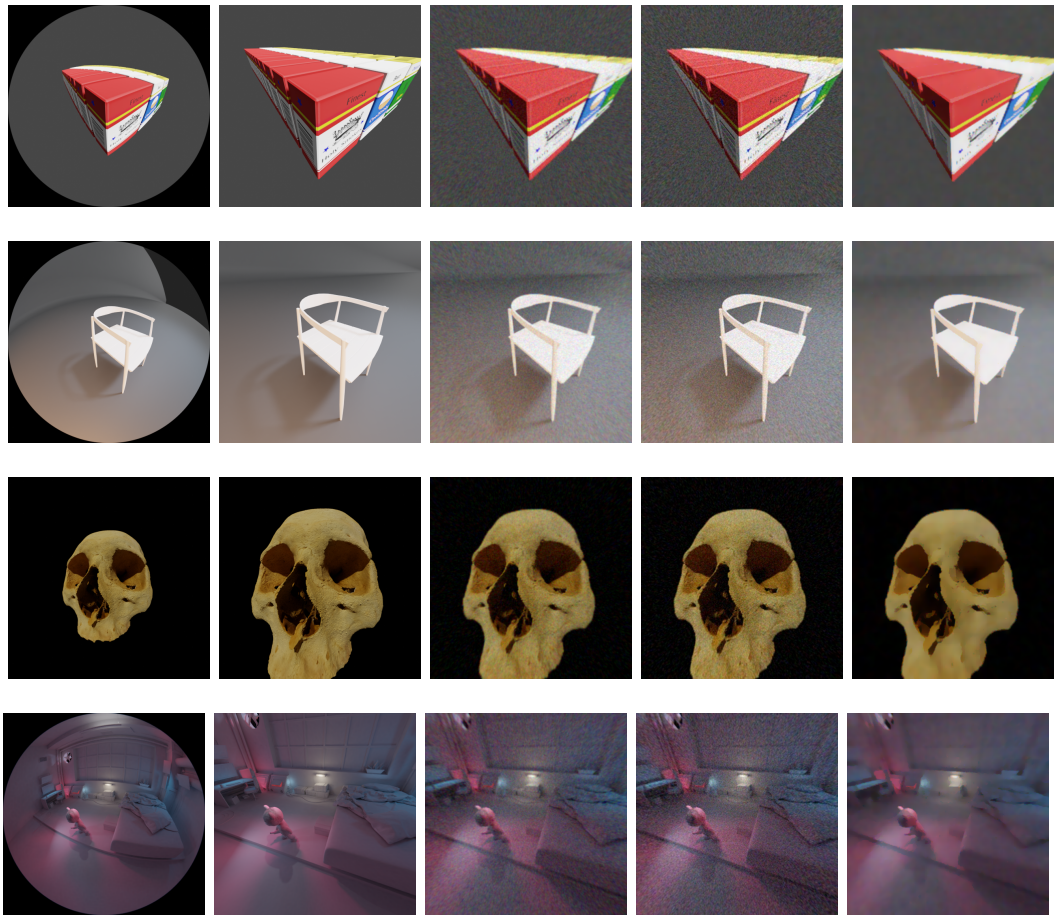
similarity (SSIM) [110], and the number of iteration of the proposed algorithm was first explored. PSNR and SSIM are two commonly used metrics in image quality assessment. Larger PSNR and SSIM mean better image perceptual quality. PSNR is a metric that computed with mean squared error (MSE) between the reconstructed images and the ground-truth images. Specifically, PSNR is defined as

$$\text{PSNR} = 20 \times \log_{10}(\frac{\text{MAX}_\text{I}}{\sqrt{\text{MSE}}}) \tag{5.1}$$

where MAX_I is the maximum possible pixel value of the image, and MSE is defined as

$$\text{MSE} = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [I(i,j) - K(i,j)]^2 \tag{5.2}$$

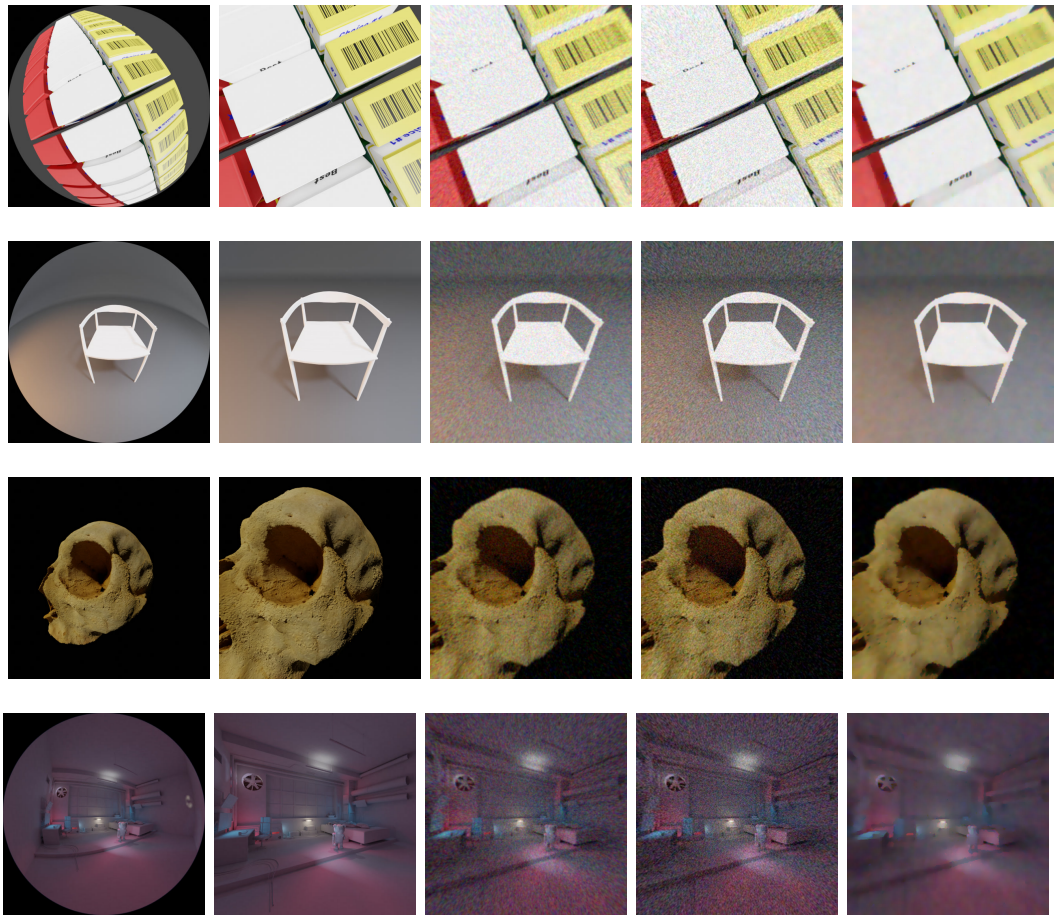
where I is a noise-free ground-truth image of size $M \times N$ and K is the



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

Figure 5.3: Demosaicking and rectification result of the in-house dataset, where the images were generated from the 3D models: `box`, `chair`, `skull` and `teddy`. The noise level was $\sigma = 10$. (a) Ground truth fisheye camera image. (b) Ground truth pinhole image. (c) Demosaicking and rectification using the bilinear method. (d) Demosaicking using HQL interpolation [74] and rectification using the bilinear method. (e) The proposed joint demosaicking / rectification method.

noisy image. SSIM is a metric that evaluates structural similarity between reconstructed images and ground-truth images, which ranges from 0 to 1. SSIM



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

Figure 5.4: Demosaicking and rectification result of the in-house dataset, where the images were generated from the 3D models: `box`, `chair`, `skull` and `teddy`. The noise level was $\sigma = 15$. (a) Ground truth fisheye camera image. (b) Ground truth pinhole image. (c) Demosaicking and rectification using the bilinear method. (d) Demosaicking using HQL interpolation [74] and rectification using the bilinear method. (e) The proposed joint demosaicking / rectification method.

between images x and y can be defined as

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1) + (2\sigma_{xy} + C_2)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{5.3}$$

where μ_x and σ_x are mean and variance of x. σ_{xy} is the covariance between x and y. C_1 and C_2 are two constants.

In this experiment, the algorithm was performed on a corrupted image with AWGN noise with a standard deviation $\sigma = 15$, and the maximum iteration number was set to 20. PSNR and SSIM in each iteration were recorded, which are shown with the blue and red lines in the figure, respectively. One can observe that both PSNR and SSIM increase rapidly in the first 8 iterations, while in the last 12 iterations, the improvement is limited. Thus, to achieved a tradeoff between performance and complexity of the proposed algorithm, 8 iterations for the proposed dataset images and 5 iterations were used for the Multi-FoV dataset images. The running time is about 1.5 minutes to process an image with a resolution of 512×512. Since the algorithm is performed on image patches, the algorithm is scalable to images with larger resolutions. μ in (3.1) was set to 1 in all settings. σ_w in (3.5) was set to 0.02 in the first iteration and 0.01 in the remaining iterations for the images from the Multi-FoV dataset, and it was set to 0.035 in the first iteration and 0.028 in the remaining iterations for the in-house dataset. The reason is that in the first iteration the image is mostly corrupted, and in the remaining iterations the iterated results are less noisy. σ_u in (3.11) was set to 1.5 for the Multi-FoV dataset and 6 for the in-house dataset. The patch size was set to 32 pixels with a stride of 28 pixels. The experiments were conducted with Matlab R2019a and a computer with a CPU of intel i7-9700T and 32G of RAM.

Table 5.1: Demosaicking and rectification performance of 25 images from scene `room` and `city` under noise level $\sigma = 15$.

	PSNR(dB)			SSIM[110]		
	Bilinear	HQL[74]	Proposed	Bilinear	HQL[74]	Proposed
room	20.76	**21.04**	20.91	0.710	0.702	**0.788**
city	24.24	24.25	**24.77**	0.550	0.557	**0.622**

Table 5.2: Demosaicking and rectification performance comparison on noise-free images from scene `box`, `chair`, `skull`, and `teddy`.

	PSNR(dB)			SSIM[110]		
	Bilinear	HQL[74]	Proposed	Bilinear	HQL[74]	Proposed
`box`	24.17	23.53	**24.21**	**0.917**	0.912	0.916
	22.64	22.10	**22.71**	**0.907**	0.902	0.906
	21.70	21.68	21.60	**0.877**	0.876	0.875
`chair`	**30.85**	30.35	30.77	**0.969**	0.967	0.967
	29.12	28.64	29.08	**0.963**	0.962	0.962
	32.35	31.86	32.16	**0.978**	0.976	0.977
`skull`	28.09	27.63	**28.15**	**0.950**	0.944	0.947
	26.20	25.56	**26.39**	**0.921**	0.909	**0.921**
	30.14	29.64	**30.21**	**0.957**	0.953	0.954
`teddy`	33.96	33.45	**33.97**	**0.958**	0.954	**0.958**
	33.06	32.56	**33.09**	**0.952**	0.947	**0.952**
	33.81	33.41	33.69	**0.962**	0.959	**0.962**

5.2 Quantitative Comparisons

The visual results for `room` and `city` are shown in Fig. 5.2. The numerical results on the Multi-FoV dataset in average SSIM [110] and PSNR are shown in Table. 5.1. In Fig. 5.2e, it can be observed that due to the proposed smoothness prior GLR in (3.1), compared with the other two methods, the results using the proposed method tend to be smoother while the boundaries in the image

Table 5.3: Demosaicking and rectification performance comparison on noisy images from scene `box`, `chair`, `skull`, and `teddy` under noise level $\sigma = 10$.

	PSNR(dB)			SSIM[110]		
	Bilinear	HQL[74]	Proposed	Bilinear	HQL[74]	Proposed
`box`	23.57	22.76	**24.00**	0.703	0.630	**0.886**
	22.21	21.52	**22.51**	0.707	0.640	**0.881**
	21.36	21.17	**21.50**	0.720	0.669	**0.857**
`chair`	28.58	27.47	**30.32**	0.751	0.670	**0.943**
	27.40	26.47	**28.77**	0.748	0.673	**0.939**
	29.37	28.16	**31.73**	0.744	0.657	**0.953**
`skull`	27.08	26.32	**27.74**	0.719	0.626	**0.892**
	25.45	24.63	**26.08**	0.760	0.686	**0.881**
	28.60	27.68	**29.70**	0.744	0.659	**0.899**
`teddy`	30.10	28.79	**32.58**	0.831	0.775	**0.935**
	29.77	28.46	**31.83**	0.828	0.771	**0.928**
	30.07	28.77	**32.34**	0.839	0.786	**0.940**

are well preserved. In addition, note that noise in the image demosaicked using the HQL method is more noticeable than the one using the bilinear method. This is because the HQL method employs filters to calculate a gradient map to enhance edges in the image. However, it may also lead to noise enhancement and image quality deterioration.

In contrast, the proposed algorithm achieves a good trade-off between edge enhancement and noise reduction. Such a conclusion is supported by Table. 5.1. Although PSNR of the HQL method outperformed the bilinear method on the Multi-FoV dataset images, its SSIM is worse than the other two methods. For the Multi-FoV dataset, PSNR of the proposed method for the `room` images is better than the bilinear method, and it is close to the PSNR of the HQL method. SSIM of the proposed method is higher than the other two methods.

Table 5.4: Demosaicking and rectification performance comparison on noisy images from scene `box`, `chair`, `skull`, and `teddy` under noise level $\sigma = 15$.

	PSNR(dB)			SSIM[110]		
	Bilinear	HQL[74]	Proposed	Bilinear	HQL[74]	Proposed
`box`	22.93	21.97	**23.85**	0.577	0.494	**0.857**
	21.74	20.90	**22.35**	0.589	0.514	**0.853**
	20.98	20.72	**21.32**	0.631	0.584	**0.825**
`chair`	26.78	25.45	**29.77**	0.604	0.512	**0.906**
	26.02	24.85	**28.38**	0.610	0.523	**0.908**
	27.27	25.81	**31.13**	0.586	0.491	**0.920**
`skull`	26.09	25.07	**27.33**	0.614	0.514	**0.863**
	24.65	23.66	**25.66**	0.682	0.603	**0.854**
	27.19	25.99	**29.22**	0.644	0.554	**0.881**
`teddy`	27.73	26.14	**31.13**	0.720	0.634	**0.884**
	27.47	25.96	**30.47**	0.716	0.636	**0.875**
	27.65	26.14	**30.90**	0.729	0.650	**0.889**

For the `city` images, PSNR and SSIM of the proposed method are better than the other two methods. The proposed method outperformed the other two methods by up to 0.52 dB in PSNR and 0.086 in SSIM, respectively.

The visual results of the in-house dataset under the noise levels $\sigma = 10$ and $\sigma = 15$ are shown in Fig. 5.3 and Fig. 5.4, respectively. Similar results can also be observed in Fig. 5.3c to 5.3e and Fig. 5.4c to 5.4e, where noise is noticeable on the image with bilinear and HQL methods. However, the proposed method is less affected by such noise. To further evaluate the performance of the proposed algorithm under different noise levels, the algorithm is executed for the noisy images under different noise levels ranging from $\sigma = 0$ to $\sigma = 20$, and the numerical results for each image are tabulated in Table. 5.2-5.5.

In Table. 5.2, it can be observed that the proposed algorithm has a similar

Table 5.5: Demosaicking and rectification performance comparison on noisy images from scene `box`, `chair`, `skull`, and `teddy` under noise level $\sigma = 20$.

	PSNR(dB)			SSIM[110]		
	Bilinear	HQL[74]	Proposed	Bilinear	HQL[74]	Proposed
`box`	22.20	21.06	**23.04**	0.481	0.405	**0.647**
	21.14	20.24	**21.82**	0.498	0.430	**0.658**
	20.55	20.14	**20.94**	0.566	0.524	**0.676**
`chair`	25.13	23.63	**27.41**	0.484	0.391	**0.686**
	24.63	23.25	**26.44**	0.498	0.407	**0.687**
	25.42	23.90	**28.07**	0.465	0.372	**0.672**
`skull`	25.05	23.74	**26.33**	0.534	0.438	**0.703**
	23.76	22.59	**24.85**	0.618	0.540	**0.739**
	25.84	24.43	**27.63**	0.576	0.480	**0.724**
`teddy`	25.74	24.05	**28.29**	0.612	0.515	**0.760**
	25.58	23.94	**27.97**	0.611	0.516	**0.753**
	25.69	23.97	**28.26**	0.622	0.528	**0.772**

performance with bilinear interpolation in noise-free settings. However, in Table. 5.3 to 5.5, it can be observed that with the increase of noise level, the performance of bilinear and HQL methods decreases significantly, while the proposed algorithm can maintain good performance across different noise settings. Specifically, in Table. 5.4, under the noise level $\sigma = 15$ it outperformed the other two competing methods by up to 4.99dB in PSNR on the images from the scene `teddy`, and up to 0.429 in SSIM on the images from scene `chair`. This is mainly because the generated images are mostly piecewise smooth images. Compared with images from the Multi-FoV dataset, these images can reap more benefit from the graph smoothness prior in (3.1).

Chapter 6

Conclusion

Fisheye cameras have drawn great interest for their wide use in virtual reality and augmented reality applications. In a conventional processing pipeline for fisheye camera images, demosaicking is first performed to interpolate missing color pixels at each pixel location, followed by rectification to correct lens distortion caused by irregular optical structure. However, during the image capturing process, as light photons are collected at each pixel location to compute an intensity value, additive noise is typically observed, which corrupts the intensity computation. Performing demosaicking and rectification in sequence means that image interpolation is executed *twice*, where at each stage, errors can accumulate, and acquisition noise at the captured pixels can smear neighbors, resulting in correlated noise, which is difficult to remove in subsequent steps.

Inspired by state-of-the-art joint demosaicking / denoising (JDD) methods that perform demosaicking and denoising in one step, in this thesis, a graph-based joint demosaicking / rectification algorithm is proposed, which performs

demosaicking to interpolate the missing pixels in the Bayer-patterned grid and corrects the fisheye camera distortion simultaneously. Specifically, a reverse mapping function is first obtained from a regular on-grid location in the rectified image to an irregular off-grid location in the camera's Bayer-patterned grid. For each pair of adjacent pixels in the rectified grid, its gradient is estimated using the pair's neighboring pixel gradients in three colors in the Bayer-patterned grid. A similarity graph is constructed based on the estimated gradients, and pixels are interpolated in the rectified grid directly via graph Laplacian regularization (GLR).

To evaluate the proposed joint demosaicking and rectification scheme, a comprehensive dataset containing ground-truth image pairs both in the fisheye camera's Bayer-patterned grid and in the rectified image grid is necessary. Unfortunately, this kind of dataset is difficult to obtain in practice. Thus, in this thesis, a synthetic image dataset that includes image pairs both on the fisheye camera's Bayer-patterned grid and the rectified image grid is constructed.

Experimental results on two fisheye camera image datasets confirm that the proposed joint demosaicking / rectification method outperformed competing methods that perform demosaicking and rectification on the fisheye camera images in separate steps.

For future work, interpolation method and gradient estimation method can be further improved. Specifically, interpolation methods with better performance, such as non-local mean methods, can be applied in this problem to further improve interpolation performance. Furthermore, the interpolation matrix $\mathbf{H}$ can be made signal-dependent, so that the interpolation can be

performed based on the image content to recreate sharper image boundaries.

In addition, in graph construction, instead of estimating gradients and computing edge weights using hand-crafted features, it is possible to extract local features automatically by leveraging convolutional neural networks (CNN). A well-trained CNN is capable of discovering sparse but relevant image features, which can potentially further improve end-to-end performance.

For the proposed dataset, compared with natural images, images in the current dataset are rendered with smooth backgrounds and without sufficient fine details. This is not common in natural images captured by real fisheye cameras, and thus these images might not be representative of natural images. For future work, more 3D models with fine details can be used for a larger-size dataset.

Bibliography

[1] Michal Aharon, Michael Elad, and Alfred Bruckstein. "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation". In: *IEEE Transactions on signal processing* 54.11 (2006), pp. 4311–4322.

[2] Mariana SC Almeida and Luis B Almeida. "Blind and semi-blind deblurring of natural images". In: *IEEE Transactions on Image Processing* 19.1 (2009), pp. 36–52.

[3] Y. Bai et al. "Graph-Based Blind Image Deblurring From a Single Photograph". In: *IEEE Transactions on Image Processing* 28.3 (Mar. 2019), pp. 1404–1418. ISSN: 1057-7149. DOI: 10.1109/TIP.2018.2874290.

[4] Yuanchao Bai et al. "Graph-based blind image deblurring from a single photograph". In: *IEEE transactions on image processing* 28.3 (2018), pp. 1404–1418.

[5] Marcelo Bertalmio et al. "Image inpainting". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 2000, pp. 417–424.

[6] F Bettonvil. "Fisheye lenses". In: *WGN, Journal of the International Meteor Organization* 33 (2005), pp. 9–14.

[7] *Blender*. `https://github.com/blender/blender`. Accessed: 2020-4-10.

[8] YY Boykov and MP Jolly. "Interactive graph cuts for optimal boundary and region segmentation of objects in nd images, 2001". In: *Proc. 8th ICCV, stranice* (), pp. 105–112.

[9] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[10] Antoni Buades, Bartomeu Coll, and J-M Morel. "A non-local algorithm for image denoising". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 2. IEEE. 2005, pp. 60–65.

[11] Antoni Buades et al. "Non local demosaicing". In: *IEEE TIP* (2007).

[12] Antoni Buades et al. "Self-similarity driven color demosaicking". In: *IEEE Transactions on Image Processing* 18.6 (2009), pp. 1192–1202.

[13] Aurélie Bugeau et al. "A comprehensive framework for image inpainting". In: *IEEE transactions on image processing* 19.10 (2010), pp. 2634–2645.

[14] Faisal Bukhari and Matthew N Dailey. "Automatic radial distortion estimation from a single image". In: *Journal of mathematical imaging and vision* 45.1 (2013), pp. 31–45.

[15] Jian-Feng Cai et al. "Blind motion deblurring from a single image using sparse approximation". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2009, pp. 104–111.

[16] Antonin Chambolle. "An algorithm for total variation minimization and applications". In: *Journal of Mathematical imaging and vision* 20.1 (2004), pp. 89–97.

[17] Yung-Hsuan Chao et al. "Pre-demosaic Graph-based Light Field Image Compression". In: *arXiv preprint arXiv:2102.07883* (2021).

[18] Priyam Chatterjee and Peyman Milanfar. "Is denoising dead?" In: *IEEE Transactions on Image Processing* 19.4 (2009), pp. 895–911.

[19] Fei Chen, Gene Cheung, and Xue Zhang. "Fast & Robust Image Interpolation using Gradient Graph Laplacian Regularizer". In: *arXiv preprint arXiv:2101.09951* (2021).

[20] G. Cheung et al. "Graph Spectral Image Processing". In: *Proceedings of the IEEE* 106.5 (May 2018), pp. 907–930. ISSN: 0018-9219. DOI: 10.1109/JPROC.2018.2799702.

[21] Gene Cheung et al. "Graph spectral image processing". In: *Proceedings of the IEEE* 106.5 (2018), pp. 907–930.

[22] Gene Cheung et al. "Robust semisupervised graph classifier learning with negative edge weights". In: *IEEE Transactions on Signal and Information Processing over Networks* 4.4 (2018), pp. 712–726.

[23] Zhixiang Chi, Xiao Shu, and Xiaolin Wu. "Joint demosaicking and blind deblurring using deep convolutional neural network". In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 2169–2173.

[24] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. 92. American Mathematical Soc., 1997.

[25] Kostadin Dabov et al. "Image denoising by sparse 3-D transform-domain collaborative filtering". In: *IEEE Transactions on image processing* 16.8 (2007), pp. 2080–2095.

[26] Adrian Davies and Phil Fennessy. *Digital imaging for photographers*. Focal Press, 2001.

[27] Weisheng Dong et al. "Compressive sensing via nonlocal low-rank regularization". In: *IEEE transactions on image processing* 23.8 (2014), pp. 3618–3632.

[28] Weisheng Dong et al. "Joint Demosaicing and Denoising with Perceptual Optimization on a Generative Adversarial Network". In: *CoRR* abs/1802.04723 (2018). arXiv: 1802.04723.

[29] Weishong Dong et al. "Joint demosaicing and denoising with perceptual optimization on a generative adversarial network". In: *arXiv preprint arXiv:1802.04723* (2018).

[30] Hilmi E Egilmez, Eduardo Pavez, and Antonio Ortega. "Graph learning from filtered signals: Graph system and diffusion kernel identification". In: *IEEE Transactions on Signal and Information Processing over Networks* 5.2 (2018), pp. 360–374.

[31] Thibaud Ehret et al. "Joint Demosaicking and denoising by fine-tuning of bursts of raw images". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 8868–8877.

[32] Andrea Eichenseer and André Kaup. "A data set providing synthetic and real-world fisheye video sequences". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, pp. 1541–1545.

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

[34] Michael Elad, Mario AT Figueiredo, and Yi Ma. "On the role of sparse and redundant representations in image processing". In: *Proceedings of the IEEE* 98.6 (2010), pp. 972–982.

[35] Sina Farsiu, Michael Elad, and Peyman Milanfar. "Multiframe demosaicing and super-resolution of color images". In: *IEEE transactions on image processing* 15.1 (2005), pp. 141–159.

[36] Sina Farsiu et al. "Fast and robust multiframe super resolution". In: *IEEE transactions on image processing* 13.10 (2004), pp. 1327–1344.

[37] *Fisheye camera model*. `https://docs.opencv.org/3.4/db/d58/group__calib3d__fisheye.html`. Accessed Jun. 27, 2019.

[38] Gerald B Folland. *Introduction to partial differential equations*. Vol. 102. Princeton university press, 1995.

[39] Giulia Fracastoro, Dorina Thanou, and Pascal Frossard. *Graph-based transform coding with application to image compression*. Tech. rep. 2019.

[40] Giulia Fracastoro et al. "Superpixel-driven graph transform for image compression". In: *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2015, pp. 2631–2635.

[41] A. Gadde, S. K. Narang, and A. Ortega. "Bilateral Filter: Graph Spectral Interpretation and Extensions". In: *IEEE International Conference on Image Processing*. Melbourne, Australia, Sept. 2013.

[42] Michaël Gharbi et al. "Deep Joint Demosaicking and Denoising". In: *ACM Trans. Graph.* 35.6 (Nov. 2016), 191:1–191:12. ISSN: 0730-0301. DOI: 10.1145/2980179.2982399.

[43] Michaël Gharbi et al. "Deep joint demosaicking and denoising". In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), pp. 1–12.

[44] Jinwook Go, Kwanghoon Sohn, and Chulhee Lee. "Interpolation using neural networks for digital still cameras". In: *IEEE Transactions on Consumer Electronics* 46.3 (2000), pp. 610–616.

[45] Ian Goodfellow et al. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016.

[46] Bart Goossens, Aleksandra Pizurica, and Wilfried Philips. "Removal of correlated noise by modeling the signal of interest in the wavelet domain". In: *IEEE transactions on image processing* 18.6 (2009), pp. 1153–1165.

[47] Bahadir K Gunturk, Yucel Altunbasak, and Russell M Mersereau. "Color plane interpolation using alternating projections". In: *IEEE transactions on image processing* 11.9 (2002), pp. 997–1013.

[48] Jonathan Harel, Christof Koch, and Pietro Perona. "Graph-based visual saliency". In: (2007).

[49] Felix Heide et al. "Flexisp: A flexible camera image processing framework". In: *ACM Transactions on Graphics (TOG)* 33.6 (2014), pp. 1–13.

[50] Janne Heikkila and Olli Silven. "A four-step camera calibration procedure with implicit image correction". In: *Proceedings of IEEE computer society conference on computer vision and pattern recognition*. IEEE. 1997, pp. 1106–1112.

[51] Theodor Heinze, Martin von Löwis, and Andreas Polze. "Joint multi-frame demosaicing and super-resolution with artificial neural networks". In: *2012 19th International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE. 2012, pp. 540–543.

[52] K. Hirakawa and T. W. Parks. "Joint demosaicing and denoising". In: *IEEE Transactions on Image Processing* 15.8 (Aug. 2006), pp. 2146–2157. ISSN: 1057-7149. DOI: 10.1109/TIP.2006.875241.

[53] Keigo Hirakawa and Thomas W Parks. "Joint demosaicing and denoising". In: *IEEE Transactions on Image Processing* 15.8 (2006), pp. 2146–2157.

[54] Tuan Ho and Madhukar Budagavi. "Dual-fisheye lens stitching for 360-degree imaging". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017, pp. 2172–2176.

[55] W. Hu, G. Cheung, and M. Kazui. "Graph-based dequantization of block-compressed piecewise smooth images". In: *IEEE Signal Processing Letters* 23.2 (2015), pp. 242–246.

[56] W. Hu, G. Cheung, and A. Ortega. "Intra-prediction and generalized graph Fourier transform for image coding". In: *IEEE Signal Processing Letters* 22.11 (2015), pp. 1913–1917.

[57] Wei Hu et al. "Graph-based joint denoising and super-resolution of generalized piecewise smooth images". In: *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2014, pp. 2056–2060.

[58] Hui-Sung Jeong and Tae-Hwan Kim. "An efficient processor for joint barrel distortion correction and color demosaicking". In: *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2015, pp. 1782–1785.

[59] Daniel Khashabi et al. "Joint demosaicing and denoising via learned non-parametric random fields". In: *IEEE Transactions on Image Processing* 23.12 (2014), pp. 4968–4981.

[60] Hyungtae Kim et al. "Fisheye lens-based surveillance camera for wide field-of-view monitoring". In: *2015 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE. 2015, pp. 505–506.

[61] Tae-Hwan Kim. "An Efficient Barrel Distortion Correction Processor for Bayer Pattern Images". In: *IEEE Access* 6 (2018), pp. 28239–28248.

[62] Teresa Klatzer et al. "Learning joint demosaicing and denoising based on sequential energy minimization". In: *2016 IEEE International Conference on Computational Photography (ICCP)*. IEEE. 2016, pp. 1–11.

[63] Shiming Lai et al. "Real-time distortion correction of fish-eye lens based on Bayer image signal". In: *Optical Review* 21.2 (2014), pp. 162–173.

[64] Yan Nei Law et al. "A semisupervised segmentation model for collections of images". In: *IEEE transactions on Image processing* 21.6 (2012), pp. 2955–2968.

[65] Yann LeCun, Yoshua Bengio, et al. "Convolutional networks for images, speech, and time series". In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.

[66] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[67] Anat Levin et al. "Seamless image stitching in the gradient domain". In: *European Conference on Computer Vision*. Springer. 2004, pp. 377–389.

[68] Guangcan Liu et al. "Robust recovery of subspace structures by low-rank representation". In: *IEEE transactions on pattern analysis and machine intelligence* 35.1 (2012), pp. 171–184.

[69] Lin Liu et al. "Joint demosaicing and denoising with self guidance". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2240–2249.

[70] X. Liu et al. "Random Walk Graph Laplacian-Based Smoothness Prior for Soft Decoding of JPEG Images". In: *IEEE Transactions on Image Processing* 26.2 (Feb. 2017), pp. 509–524. ISSN: 1057-7149. DOI: 10.1109/TIP.2016.2627807.

[71] Xinhao Liu, Masayuki Tanaka, and Masatoshi Okutomi. "Practical signal-dependent noise parameter estimation from a single noisy image". In: *IEEE Transactions on Image Processing* 23.10 (2014), pp. 4361–4371.

[72] Florian Luisier, Thierry Blu, and Michael Unser. "Image denoising in mixed Poisson–Gaussian noise". In: *IEEE Transactions on image processing* 20.3 (2010), pp. 696–708.

[73] Hiêp Quang Luong et al. "A primal-dual algorithm for joint demosaicking and deconvolution". In: *2012 19th IEEE International Conference on Image Processing*. IEEE. 2012, pp. 2801–2804.

[74] H. S. Malvar, Li-wei He, and R. Cutler. "High-quality linear interpolation for demosaicing of Bayer-patterned color images". In: *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 3. May 2004, pp. iii–485. DOI: 10.1109/ICASSP.2004.1326587.

[75] Henrique S Malvar, Li-wei He, and Ross Cutler. "High-quality linear interpolation for demosaicing of Bayer-patterned color images". In: *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 3. IEEE. 2004, pp. iii–485.

[76] Tomer Michaeli and Michal Irani. "Blind deblurring using internal patch recurrence". In: *European conference on computer vision*. Springer. 2014, pp. 783–798.

[77] Peyman Milanfar. "A tour of modern image filtering: New insights and methods, both practical and theoretical". In: *IEEE signal processing magazine* 30.1 (2012), pp. 106–128.

[78] Peyman Milanfar. *Super-resolution imaging*. CRC press, 2017.

[79] Martin Fodslette Moller. "A scaled conjugate gradient algorithm for fast supervised learning". In: *Neural Networks* 6.4 (1993), pp. 525–533.

[80] Seonghyeon Nam et al. "A holistic approach to cross-channel image noise modeling and its application to image denoising". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1683–1691.

[81] Michael K Ng, Guoping Qiu, and Andy M Yip. "Numerical methods for interactive multiple-class image segmentation problems". In: *International Journal of Imaging Systems and Technology* 20.3 (2010), pp. 191–201.

[82] Nhat Nguyen, Peyman Milanfar, and Gene Golub. "A computationally efficient superresolution image reconstruction algorithm". In: *IEEE transactions on image processing* 10.4 (2001), pp. 573–583.

[83] Antonio Ortega et al. "Graph signal processing: Overview, challenges, and applications". In: *Proceedings of the IEEE* 106.5 (2018), pp. 808–828.

[84] J. Pang and G. Cheung. "Graph Laplacian Regularization for Image Denoising: Analysis in the Continuous Domain". In: *IEEE Transactions on Image Processing* 26.4 (Apr. 2017), pp. 1770–1785. ISSN: 1057-7149. DOI: 10.1109/TIP.2017.2651400.

[85] Sylvain Paris et al. *Bilateral filtering: Theory and applications*. Now Publishers Inc, 2009.

[86] Bo Peng, Lei Zhang, and David Zhang. "A survey of graph theoretical approaches to image segmentation". In: *Pattern recognition* 46.3 (2013), pp. 1020–1038.

[87] Benjamin Petry and Jochen Huber. "Towards effective interaction with omnidirectional videos using immersive virtual reality headsets". In: *Proceedings of the 6th Augmented Human International Conference*. 2015, pp. 217–218.

[88] Nikolay N Ponomarenko et al. "A method for blind estimation of spatially correlated noise characteristics". In: *Image Processing: Algorithms and Systems VIII*. Vol. 7532. International Society for Optics and Photonics. 2010, p. 753208.

[89] William H Press. *Numerical Recipes in C: C Version*. Cambridge university press, 1992.

[90] Wenqi Ren et al. "Image deblurring via enhanced low-rank prior". In: *IEEE Transactions on Image Processing* 25.7 (2016), pp. 3426–3437.

[91] Leonid I Rudin, Stanley Osher, and Emad Fatemi. "Nonlinear total variation based noise removal algorithms". In: *Physica D: nonlinear phenomena* 60.1-4 (1992), pp. 259–268.

[92] Richard M Satava. "Virtual reality, telesurgery, and the new world order of medicine". In: *Journal of Image Guided Surgery* 1.1 (1995), pp. 12–16.

[93] Godwin Shen et al. "Edge-adaptive transforms for efficient depth map coding". In: *28th Picture Coding Symposium*. IEEE. 2010, pp. 566–569.

[94] Huanfeng Shen and Liangpei Zhang. "A MAP-based algorithm for destriping and inpainting of remotely sensed images". In: *IEEE Transactions on Geoscience and Remote Sensing* 47.5 (2008), pp. 1492–1502.

[95] David I Shuman et al. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *IEEE signal processing magazine* 30.3 (2013), pp. 83–98.

[96] D. I. Shuman et al. "The Emerging Field of Signal Processing on Graphs: Extending High-dimensional Data Analysis to Networks and Other Irregular Domains". In: *IEEE Signal Processing Magazine*. Vol. 30. 3. May 2013, pp. 83–98.

[97] David Strong and Tony Chan. "Edge-preserving and scale-dependent properties of total variation regularization". In: *Inverse problems* 19.6 (2003), S165.

[98] Weng-tai Su et al. "Graph Neural Net Using Analytical Graph Filters and Topology Optimization for Image Denoising". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 8464–8468.

[99] Jack Tan, Gene Cheung, and Rui Ma. "360-degree virtual-reality cameras for the masses". In: *IEEE multimedia* 25.1 (2018), pp. 87–94.

[100] Thorsten Thormählen, Hellward Broszio, and Ingolf Wassermann. "Robust line-based calibration of lens distortion from a single view". In: *Mirage 2003* (2003), pp. 105–112.

[101] C. Tomasi and R. Manduchi. "Bilateral Filtering for Gray and Color Images". In: *ICCV*. Bombay, India, 1998.

[102] Carlo Tomasi and Roberto Manduchi. "Bilateral filtering for gray and color images". In: *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*. IEEE. 1998, pp. 839–846.

[103] Mikhail L Uss et al. "Local signal-dependent noise variance estimation from hyperspectral textural images". In: *IEEE Journal of Selected Topics in Signal Processing* 5.3 (2011), pp. 469–486.

[104] Patrick Vandewalle et al. "Joint demosaicing and super-resolution imaging from a set of unregistered aliased images". In: *Digital Photography III*. Vol. 6502. International Society for Optics and Photonics. 2007, 65020A.

[105] Richard S Varga. *Gervsgorin and his circles*. Vol. 36. Springer Science & Business Media, 2010.

[106] Pranav Verma et al. "Splatty-a Unified Image Demosaicing and Rectification Method". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pp. 786–795.

[107] Sviatoslav Voloshynovskiy, Oleksiy Koval, and Thierry Pun. "Image denoising based on the edge-process model". In: *Signal Processing* 85.10 (2005), pp. 1950–1969.

[108] Huy Vu, Gene Cheung, and Yonina C Eldar. "Unrolling of Deep Graph Total Variation for Image Denoising". In: *arXiv preprint arXiv:2010.11290* (2020).

[109] Aiqi Wang, Tianshuang Qiu, and Longtan Shao. "A simple method of radial distortion correction with centre of distortion estimation". In: *Journal of Mathematical Imaging and Vision* 35.3 (2009), pp. 165–172.

[110] Z. Wang et al. "Image Quality Assessment: From Error Visibility to Structural Similarity". In: *IEEE Transactions on Image Processing*. Vol. 13, no.4. Aug. 2005, pp. 600–612.

[111] Li Xu, Shicheng Zheng, and Jiaya Jia. "Unnatural l0 sparse representation for natural image deblurring". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, pp. 1107–1114.

[112] Xuan Xu, Yanfang Ye, and Xin Li. "Joint demosaicing and super-resolution (JDSR): network design and perceptual optimization". In: *IEEE Transactions on Computational Imaging* (2020).

[113] Jianchao Yang et al. "Image super-resolution via sparse representation". In: *IEEE transactions on image processing* 19.11 (2010), pp. 2861–2873.

[114] Lu Yuan et al. "Image deblurring with blurred/noisy image pairs". In: *ACM SIGGRAPH 2007 papers*. 2007, 1–es.

[115] Jin Zeng et al. "Deep graph Laplacian regularization for robust denoising of real images". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.

[116] L. Zhang, X. Wu, and D. Zhang. "Color Reproduction From Noisy CFA Data of Single Sensor Digital Cameras". In: *IEEE Transactions on Image*

Processing 16.9 (Sept. 2007), pp. 2184–2197. ISSN: 1057-7149. DOI: 10.1109/TIP.2007.901807.

[117] Lei Zhang and Xiaolin Wu. "Color demosaicking via directional linear minimum mean square-error estimation". In: *IEEE Transactions on Image Processing* 14.12 (2005), pp. 2167–2178.

[118] Lei Zhang et al. "Color demosaicking by local directional interpolation and nonlocal adaptive thresholding". In: *Journal of Electronic imaging* 20.2 (2011), p. 023016.

[119] Lei Zhang et al. "PCA-based spatially adaptive denoising of CFA images for single-sensor digital cameras". In: *IEEE transactions on image processing* 18.4 (2009), pp. 797–812.

[120] Zhengyou Zhang. "A flexible new technique for camera calibration". In: *IEEE Transactions on pattern analysis and machine intelligence* 22.11 (2000), pp. 1330–1334.

[121] Ruofan Zhou, Radhakrishna Achanta, and Sabine Süsstrunk. "Deep residual network for joint demosaicing and super-resolution". In: *Color and imaging conference*. Vol. 2018. 1. Society for Imaging Science and Technology. 2018, pp. 75–80.

[122] Yuqian Zhou et al. "When awgn-based denoiser meets real noises". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 13074–13081.

[123] Zichao Zhang et al. "Benefit of large field-of-view cameras for visual odometry". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. May 2016, pp. 801–808. DOI: 10.1109/ICRA.2016.7487210.