ÉCOLE CENTRALE DE NANTES

MASTER CORO-IMARO
"CONTROL AND ROBOTICS"

2019 / 2020

Master Thesis Report

Presented by

Zihua XU

28th August 2020

Omnidirectional Image Inpainting with Deep Generative Models

Jury

Evaluators:	Olivier KERMORGANT	Assistant Professor (ECN)
	Philippe MARTINET	Research Director (INRIA)
	Patrick RIVES	Research Director (INRIA)
Supervisor:	Renato MARTINS	Postdoctoral Research Fellow (INRIA)

Laboratory: Laboratoire des Sciences du Numérique de Nantes LS2N

Abstract

Omnidirectional and panoramic images encode interesting properties to several computer vision and robotics problems, such as visibility invariance to camera rotation. However, they are still often obtained with rigs of cameras and therefore are subjected to missing image regions, and to visual artifacts coming from composition and stitching operations done for creating the panoramas. Therefore, for some applications, automatic image editing is required to improve the quality of these images. Image manipulation techniques, that restore and complete missing parts of images, are named inpainting, which witnessed important recent advances from generative adversarial networks on image-to-image translation problems. But most of these techniques are designed to perspective flat images, and therefore are not adapted to omnidirectional images such as fisheye, catadioptric or equirectangular images. The main objective of this master thesis is to design a generative inpainting approach for omnidirectional images. The state-of-the-art inpainting algorithm "Globally and Locally Consistent Image Completion" (GLCIC) algorithm was selected as our backbone generative model. We adopted this model for filling missing regions in omnidirectional images. The performance of our inpainting model is evaluated in a bottom-up complexity scheme: reconstruction, denoising and generative inpainting using a set of omnidirectional images taken from indoor scenes. The evaluation is done with structural and perceptual metrics such as the mean squared error (MSE), structural similarity and learned perceptual image patch similarity (LPIPS) to jointly evaluate the inpainting results. The obtained results show the potential of generative models for inpainting omnidirectional images.

Acknowledgements

I would like to thank everyone who have helped me during my master thesis process. First of all I would like to express my deep and sincere thanks to my supervisor Dr. Renato Martins for providing me with the opportunity to pursue this thesis, monitoring our project and providing valuable inputs with attention and care. It's a pleasure to work with my supervisor. He not only help me to solve specific problems but also give me some ideas on how to do research step by step in a scientific way. The way to grasp the big picture of the problem, to formalize the problem, to find valuable literature, to show the results properly, to write the manuscript in a scientific way, are all very precious experience for me. His supports and encouragement during my frustrated period also help me a lot. I really appreciate his patience and kindness all along the journey. I would like to thanks INRIA Sophia Antipolis to provide me this excellent environment and atmosphere for my work, and all my colleges in CHORALE group who always give me lots of supports and pleasures during these six months. Finally on a personal note, I would like to thank my family for their emotional support and care during my study. I also would like to thank Ayane Sakura, Maaya Uchida, Reina Ueda, Taneda Risa, Minami Tanaka. I was cured by their voice and cuteness during the hard time.

Notations

P	3D point in $\mathbb{R}^3$
P_s	Point on the unit sphere surface of the mirror frame
P_ϵ	Point on the unit sphere and in the camera frame
p	Point on image plane
x_s, y_s, z_s	Coordinates on the unit sphere surface
x_m, y_m	Coordinates on the normalized image plane
m	Point on the normalized image plane
K	Camera intrinsic matrix
π	Projection function from 3D point to 2D point
Ω	Set of valid 3D projection points
Θ	Set of valid 2D projection points
ρ	Distance between one 2D point and image center
λ	Depth scale
T	Rigid transformation matrix
u''	Perfect project point
u'	Actual project point
A	Affine matrix
V	Calibration parameters
x_d, y_d	Coordinates of desired point
θ_k, ϕ_k	Coordinates of equirectangular image
μ	mean value
σ	Standard deviation
e_θ, e_ϕ	Unit vector on the unit sphere
Y_{lm}	Spherical Harmonics
I	Image intensity function
$\hat{I}_{lm}$	Coefficients of the spherical Fourier transform of the signal I
R	Rotation matrix
$d^l_{km}(\beta)$	Wigner d-function
$D^l_{k,m}(g)$	Wigner D-function
$(\boldsymbol{I} \star \boldsymbol{H})(g)$	spectral convolution
$G(t)_{lm}$	Coefficients of spherical Gaussian
$SO(3)$	3D rotation group
S^2	Sphere space

Abbreviations

FOV	Field of View
UCM	Unified Camera Model
EUCM	Extended Unified Camera Model
DSCM	Double Sphere Camera Model
MSE	Mean Squared Error
SSIM	Structural Similarity Index Metric
LPIPS	Learned Perceptual Image Patch Similarity
ALP	Associated Legendre Polynomials
SH	Spherical Harmonics
CNNs	Convolutional Neural Networks
GANs	Generative Adversarial Networks
GLCIC	Globally and Locally Consistent Image Completion
MNIST	Modified National Institute of Standards and Technology dataset
BCE	Binary Cross Entropy

List of Figures

1.1 Examples of recent generative inpainting approaches. . . . 1
1.2 Outdoor panoramic image building from stitching perspective images. . . . 3
1.3 Indoor panoramic image building from stitching fisheye images. . . . 3

2.1 The projection schematic of UCM and EUCM models. . . . 6
2.2 Spectral convolution operators. . . . 6
2.3 Comparison between the regular neighbors and the spherical ones. . . . 7
2.4 The blurring effect when completing large missing regions with diffusion-based inpainting. . . . 7
2.5 Removing objects with patch-based inpainting. . . . 8
2.6 The structure of the generator in DCGAN. . . . 10
2.7 Facial images generated by a VAE and a DCGAN. . . . 11
2.8 The structure of the global and local consistent image completion network. . . . 11

3.1 A simple image convolution example. . . . 14
3.2 Sobel operators. . . . 14
3.3 A CNN architecture to classify handwritten digits. . . . 15
3.4 Transposed convolution layer. . . . 15
3.5 The omnidirectional cameras. . . . 16
3.6 The unified camera projection model. . . . 17
3.7 Taylor camera model. . . . 19
3.8 The visual results of the calibration of UCM. . . . 20
3.9 The reprojected points and pixels on the checkerboards. . . . 21
3.10 The (θ, ϕ) equirectangular image project from omnidirectional image. . . . 22
3.11 Back-projected results. . . . 23
3.12 Examples of back-projected images with wrong parameters. . . . 24
3.13 Histogram of the original images and the back-projected ones. . . . 25
3.14 Spherical coordinates and local basis for defining pixel neighbourhood. . . . 26
3.15 Spherical neighbours project on the equirectangular plane. . . . 28
3.16 Visualization of spherical harmonics. . . . 29
3.17 The coordinate system of spherical image. . . . 30
3.18 Reconstructed spherical image with a different order of spherical harmonics. . . . 31
3.19 The histogram of the original image and the reconstructed ones. . . . 33
3.20 The equirectangular image with a red circle that rotated by different angles. . . . 34
3.21 The circle image rotated by the rotation matrices generated by the recursive method. . . . 35
3.22 The spherical Gaussian smoothing results. . . . 36
3.23 Examples of flat Gaussian convolution and spherical Gaussian convolution. . . . 36

4.1 Structure of the convolutional autoencoder and the adapted one. . . . 41

4.2 Comparing between flat digits and spherical digits. . . . 42
4.3 Training losses for reconstructing spherical digits. . . . 43
4.4 Reconstruction of the spherical digits. . . . 43
4.5 The structure of the autoencoders. . . . 44
4.6 Data augmentation of the spherical image. . . . 45
4.7 Training losses of the original autoencoder and the adapted one. . . . 46
4.8 Part I: Reconstruction results of both the original generator and then adapted one to equirectangular images. . . . 46
4.9 Part II Reconstruction results of both the original model and the adapted one to equirectangular images. . . . 47
4.10 Part I: Results of inpainting for the missing regions. . . . 49
4.11 Part II: Results of inpainting for the missing regions. . . . 50

5.1 The structure of the autoencoders. . . . 54
5.2 The structure of discriminators. . . . 55
5.3 Some indoor images from the dataset SUN360. . . . 56
5.4 The results of the inpainting for the omnidirectional images with the original generative model. . . . 58
5.5 The five boxes indicate the inpainting areas in each image. . . . 58
5.6 Training losses of the generative models. . . . 59
5.7 Accuracy results of the discriminators. . . . 60
5.8 Part I: Inpainting with indoor images. . . . 61
5.9 Part II: Inpainting with indoor images. . . . 62

List of Tables

2.1 Comparison between traditional inpainting and the inpainting from deep generative models. 9

3.1 Different similarity metrics applied on the images of Figure 3.11 24
3.2 Similarity metrics applied on the images in Figure 3.18 32

4.1 Evaluation of reconstruction (digit numbers) with the SSIM and LPIPS metrics. 42
4.2 Evaluation of reconstruction (real images) with the SSIM and LPIPS metrics. . 48
4.3 Evaluation of inpainting results with MSE and LPIPS. 51

5.1 Evaluation of inpainting results with MSE and LPIPS. 63

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Problem Statement and Main Objectives . . . 2
- 1.3 Thesis Outline . . . 2

2 Related Work **5**
- 2.1 Omnidirectional Vision . . . 5
  - 2.1.1 Convolution for Omnidirectional Images . . . 5
- 2.2 Inpainting Approaches . . . 7
- 2.3 Inpainting from Deep Generative Models . . . 9
  - 2.3.1 Variational Autoencoders . . . 10
  - 2.3.2 Generative Adversarial Networks . . . 10

3 Omnidirectional Image Processing **13**
- 3.1 Introduction to Image Processing . . . 13
- 3.2 Introduction to Omnidirectional Images . . . 15
  - 3.2.1 Camera Projection Models . . . 16
  - 3.2.2 Camera Calibration . . . 19
  - 3.2.3 Calibration Results . . . 20
- 3.3 Omnidirectional Image Mapping to the Unit Sphere . . . 21
  - 3.3.1 Results of the Equirectangular Projection . . . 22
- 3.4 Omnidirectional Image Processing Methods . . . 24
  - 3.4.1 Redefining Pixel Neighboring and Projection Approaches . . . 26
- 3.5 Spherical Image Processing with Spectral Analysis . . . 27
  - 3.5.1 Orthogonal Basis Functions . . . 28
  - 3.5.2 Spherical Harmonics . . . 29
  - 3.5.3 Properties of Spherical Harmonics Functions . . . 32
  - 3.5.4 Rotation of Spherical Harmonics . . . 32
  - 3.5.5 Spectral Convolution . . . 33
  - 3.5.6 Results of Convolution with Spherical Gaussian . . . 35
- 3.6 Conclusions . . . 37

4 Inpainting of Panoramas from Reconstruction Networks **39**
- 4.1 Inpainting Problem Formulation . . . 39
- 4.2 Reconstruction for Omnidirectional Images . . . 40
  - 4.2.1 Reconstruction of Spherical Digits . . . 40
  - 4.2.2 Reconstruction of Real Panoramas . . . 42
- 4.3 Inpainting from Denoising with Convolutional Autoencoder . . . 48
  - 4.3.1 Architecture and Loss Function . . . 48
  - 4.3.2 Results of Inpainting . . . 49

4.3.3 Quantitative Evaluation 51

5 Inpainting with Generative Models **53**
5.1 Introduction to Generative Models 53
5.2 Generative Inpainting Model 54
5.2.1 Losses Functions 54
5.3 Datasets 55
5.3.1 Real Indoor Omnidirectional Dataset 56
5.4 Training Strategy 56
5.5 Results of Generative Inpainting 56
5.5.1 Evaluation 62

6 Conclusions **65**

References **67**

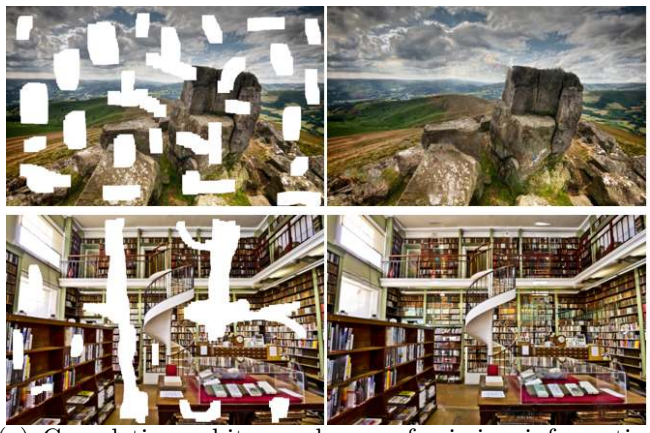
Chapter 1

Introduction

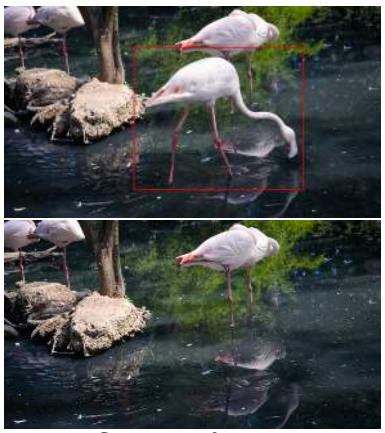
1.1 Motivation

Images are everywhere. As entertainment, animation, and augmented reality developments are more present in everyday devices and applications, the requirement of high-quality images also increased. However, some images do not present the expected visual quality for various reasons, including scratches, overlaid text, missing parts, or undesired objects. The techniques to handle these problems are called inpainting. Image inpainting consists of enhancing the quality of images by the completion of damaged or missing information. Inpainting can also be used in image editing situations where we want to remove undesired objects. Due to their importance, many automated inpainting techniques have been developed to reduce human manual work, which even can generate more plausible images than the ones edited by a graphic designer.

Encouraging recent inpainting results have been recently developed with generative adversarial networks (e.g., [21, 53]). Generative inpainting uses adversarial generative networks to directly synthesize the missing regions of images. Examples of the application of generative inpainting models are for completing missing areas, and removing objects are shown in Figure 1.1. Note that inpainting needs to complete the cropped image plausibly and provide a realistic appearance. But most generative inpainting techniques are designed for perspective images. This type of image is simpler to represent and easier to manipulate than omnidirectional images. Many systems such as autonomous vehicles, drones, and virtual reality equipments often use omnidirectional images, since panoramic images can provide much larger FOV to help the algorithms make better decisions.



(a) Completing arbitrary shapes of missing information in photos. Image courtesy of [21].



(b) Removing a flamingo from an image with inpainting. Image courtesy of [53].

Figure 1.1: Examples of recent generative image and video inpainting approaches.

Although the intrinsic advantages from omnidirectional images, they are more challenging to model than images from pinhole perspective camera models. Notably, omnidirectional images have a non-uniform resolution in the image plane. Therefore modeling the visual information from non-uniform resolution and distorted objects is an intrinsic difficulty of these images. The works of [16], [42] and [47], proposed camera models and calibrations to some of these camera systems. These techniques help us in representing and processing the omnidirectional images. Furthermore, omnidirectional images also require adapted image processing methods, such as discussed in the works from [13] and [19]. The conventional inpainting methods like diffusion-based inpainting [4], texture synthesis [57] only uses internal information from the single input image to fill the missing regions. But these methods require appropriate information to be contained in the input image, like similar pixels, structures, or patches. This assumption is hard to satisfy if the missing region is vast and possibly of arbitrary shape, as shown in the examples depicted in Figure 1.1. Recent approaches are the combining the external information and the internal one to perform inpainting. The main goal of this project is to explore these recent advances to design an image inpainting approach to panoramic images. In this report, we present the above-mentioned techniques and show how to combine them to develop a generative inpainting model for inpainting omnidirectional images.

1.2 Problem Statement and Main Objectives

Our main objective is to restore missing or corrupted information as shown in Figures 1.2 and 1.3. The missing or damaged information cause difficulties when directly using these images in the applications. Therefore, our project endeavors to design an inpainting approach to reduce these problems.

The pixel distribution in omnidirectional images is not spatially invariant due to the distortions. Therefore, processing methods for perspective images are not applicable for omnidirectional images. In this thesis, we mainly focus on the convolution, since it is one of the essential components of the generative models. There are two convolutional methods for panoramas. One is the equirectangular convolution [45], while the other one is the spectral convolution [9]. We want to design a generative adversarial network with these convolutional methods to perform inpainting on the panoramas. However, generative adversarial networks are sensitive to the architecture, parameters and are notoriously hard to train.

To design our generative inpainting technique, we divide this goal into two main steps. First of all, we need to represent omnidirectional images with a convenient camera model, which can project the panoramas to the unit sphere, since the convolutions for panoramas process the images in this domain. And then, we establish our baseline in a bottom-up complexity scheme: reconstruction, denoising, and generative inpainting models using a set of omnidirectional images taken from indoor scenes. Finally, we evaluate the inpainting results with several evaluation metrics to compare the performance between an inpainting network model built with classical processing operators and the proposed adapted model.

1.3 Thesis Outline

The rest of this report is divided into four chapters. Chapter 2 briefly presents recent developed image inpainting techniques. We introduce the state-of-the-art generative inpainting techniques, which are used as the backbone of our inpainting method. Chapter 3 demonstrates how to represent and process omnidirectional images adequately. We first show and discuss typical large FOV models to represent omnidirectional images. It also contains the fundamental theoretical background of signal processing operators for omnidirectional images. We will stress

Figure 1.2: Outdoor panoramic image building from stitching perspective images. As can be noticed this panorama has missing information indicated by black regions, which we would like to complete.

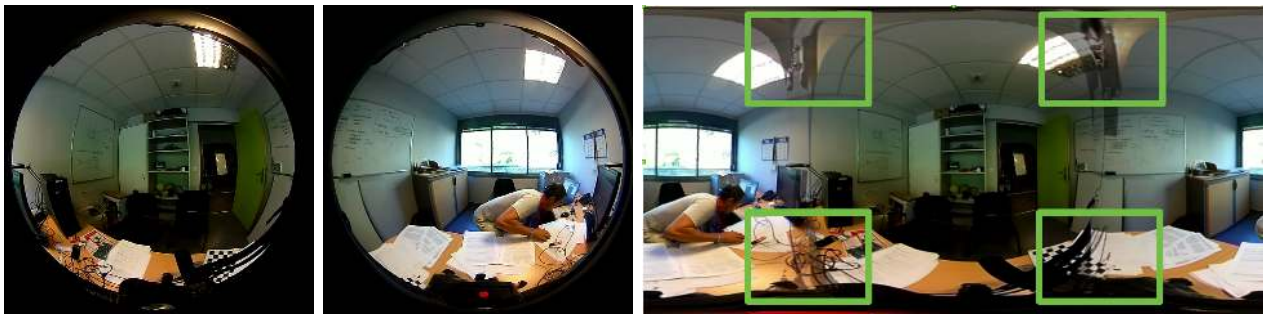


Figure 1.3: Indoor panoramic image building from stitching fisheye images. The images are corrupted by the mounting camera device that appears in the FOV (displayed in the external border of the two fisheye images). The green boxes indicate the places that need inpainting.

in this part the advantages provided by large FOV images, but also the intrinsic difficulties that need to be addressed to explore them. We also present evaluation for critical image processing operators such as calibration of the omnidirectional camera, equirectangular convolution, and spherical convolution.

Chapter 4 presents the problem of image inpainting with neural networks. We first design a convolutional autoencoder to perform the reconstruction and inpainting on omnidirectional images. By evaluating the results of this model, we can assess the performance of the equirectangular convolution and help us to understand the required modifications on a more complex generative network. Chapter 5 introduces our modify generative inpainting network. We demonstrate and discuss the inpainting results. We also evaluate these results with different evaluation metrics to assess the performance of our generative inpainting model.

Chapter 2

Related Work

Omnidirectional images contain much larger field of view than perspective images. Therefore, they are more and more applied in different fields, such as robotics, and virtual reality. The requirements of inpainting for panoramas also increased. Image inpainting is one of the techniques of image restoration. This technique fills missing or damaged parts of the image reasonably (after inpainting it should be hard to recognize from visual inspect where are the filled regions). We can also use inpainting to manipulate images, such as to remove some undesired parts in the images by marking these parts as missing. In this chapter, we present the related works of extracting information from the panoramas. We also present recent related works on image inpainting, which inspired us to design our inpainting method for omnidirectional images.

2.1 Omnidirectional Vision

To correctly extract the information from omnidirectional images, we need to build the relationship between the pixels and the 3D points in the world. In [16], they proposed a unified camera model (UCM), which uses the unit sphere surface to connect the 3D points and the pixels. The projection schematic of this model is shown in Figure 2.1. This model has a closed-form solution for the projection from pixels to the 3D points. Furthermore, this model can represent both catadioptric and fisheye images. Based on this model, [26] proposed the extended unified camera model (EUCM). In this model, the 3D points are not projected into the unit sphere surface but into a symmetric ellipsoid around the Z axis. The projection schematic of EUCM also is shown in Figure 2.1. EUCM can represent all possible quadratic surfaces, which expanded the scope of UCM's application. Recently, the double sphere camera model (DSCM) was proposed by [47]. DSCM fits well cameras with fisheye lenses, has a closed-form inverse, and does not require computationally expensive computations for the mapping.

2.1.1 Convolution for Omnidirectional Images

There are many types of image processing operators, but we mainly focus on convolution. Because convolutions are the main components of recent neural network models. Since the pixel distribution of omnidirectional images is not spatially invariant, we cannot directly apply the perspective image convolution for panoramas. Thanks to the camera models, such as UCM and EUCM, we can correctly project the pixels to the unit sphere.

The work [9] proposed a spherical CNN that uses Fourier transform to change the image and convolution kernel to the frequency domain. After performing convolution in the frequency domain, the inverse Fourier transform is used to obtain the final results. The processing procedures are shown in Figure 2.2. This method can handle well the spherical signal. However, it is not easy to apply to the inpainting context. When using Fourier transform to reconstruct the signal, the higher bandwidth, the more information is kept. The high bandwidth signal

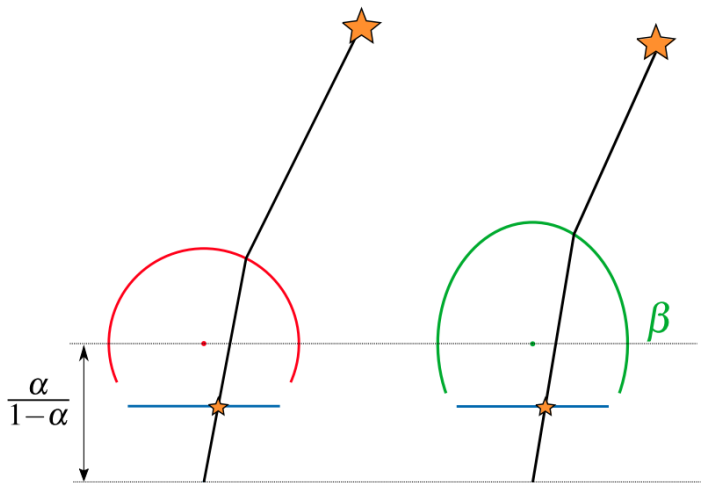


Figure 2.1: The projection schematic of UCM and EUCM models. The big stars present the 3D points, while the small ones present the pixels. Firstly, a 3D point is projected onto a unit sphere and then projected onto the image plane of the pinhole camera shifted by $\frac{\alpha}{1-\alpha}$ from the center of the sphere. In the EUCM, the sphere is transformed into an ellipsoid using the coefficient β. Image courtesy of [47].

requires high computing resources. To obtain reconstruction results of high-quality images, this method has expensive computational cost.

In [45], they proposed the distortion-aware convolutional filters, which defined the spherical neighbors on the unit sphere. The comparison of the traditional neighbors selection on perspective images and the spherical ones can be seen in Figure 2.3. We can see that spherical neighbors exploit projections (red) of the sampling pattern on the tangent plane (blue), yielding neighbors' outputs which are invariant to latitudinal rotations. The distortion-aware convolutional filters just need to compute the new neighbors' positions once.

With these convolutions, we can extract the information from the panoramas and represent them into certain commonly understood representations. Based on these descriptions, we can perform different image processing tasks on panoramas, such as classification [8], matching between consecutive images of a sequence [13], visual odometry [19] and inpainting [11].

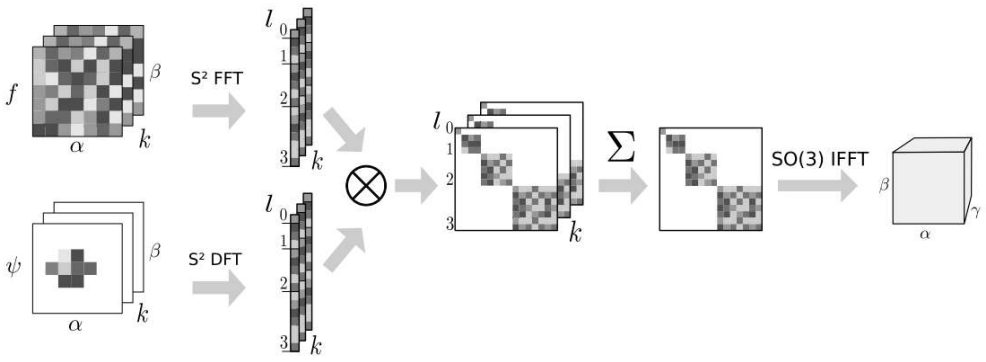


Figure 2.2: Spectral convolution operators. The signal f and the locally-supported filter ϕ are Fourier transformed, block-wise tensored, summed over input channels, and finally inverse transformed. The input signal is in the spherical coordinates α, β, and and output signal is given in SO(3) domain with ZYZ-Euler angles α, β, γ. Image courtesy of [8].

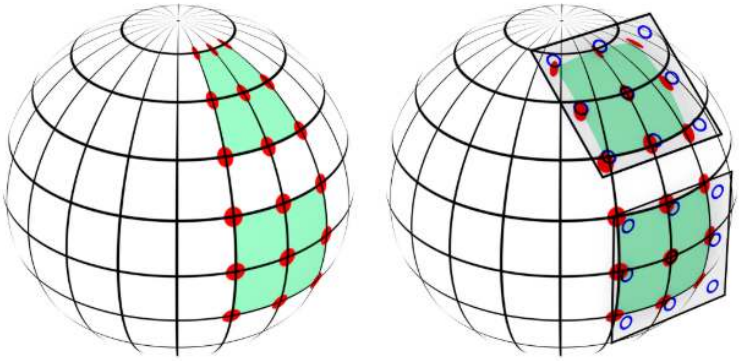


Figure 2.3: Comparing between the regular neighbors and the spherical ones. The left one is the regular sampling neighbors on the sphere, while the right one is the spherical sampling neighbors. Image courtesy of [8].

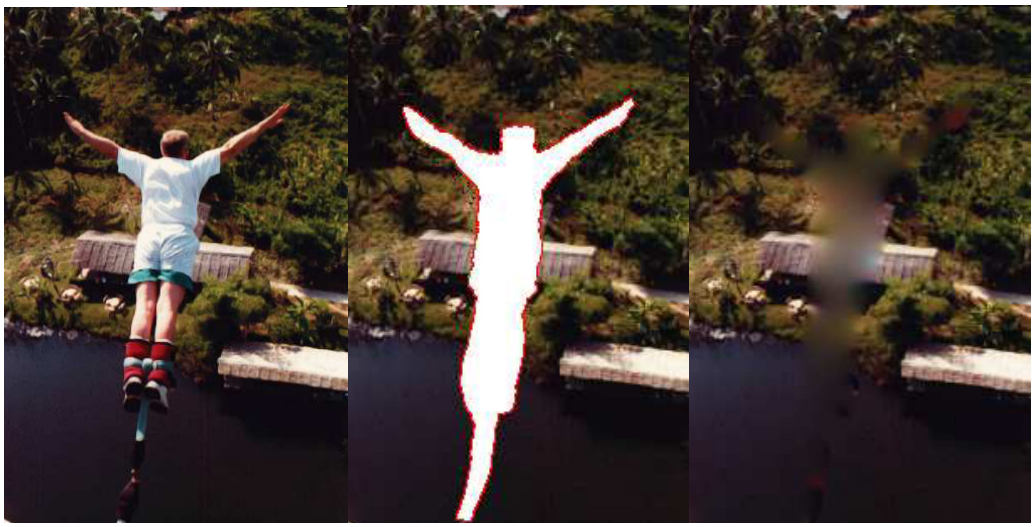
Figure 2.4: Completing huge missing area with diffusion-based inpainting. Image courtesy of [31].

2.2 Inpainting Approaches

We can divide image inpainting into several types according to the specific implementation process . We denote the inpainting techniques that can restore new samples by using the generative models as generative inpainting, while the others are denoted as traditional inpainting. A variety of different approaches have been proposed for inpainting. Diffusion-based inpainting is the first digital inpainting technique. This technique diffuses the image information from the known region into the missing region at the pixel level. Fundamentally these algorithms are based on variational methods and Partial Differential equations (PDE). PDE decides the filling order and filling information. Then the variational term maintains the smoothness of the image and eliminate the artifacts caused by PDE. For example, the propagation can be performed based on the isophote direction field [1], or using global image statistics based on the histograms of local features [30]. This method can keep the linear structure of the image. But its disadvantage is that the restored images often contain blur and lose details (especially the textures) [4], [5]. Therefore, diffusion-based inpainting is more appropriate for small, non-textured image regions. A typical example, where diffusion-based inpainting failed to complete a missing region, is shown in Figure 2.4.

Patch-based approaches were proposed to overcome the limitations of diffusion-based inpainting, . These approaches can perform more complicated image inpainting to fill large regions in natural images. Patch-based image completion was first proposed for texture syn-

(a) Removing a road sign.



(b) Removing a human.

Figure 2.5: Removing objects with patch-based inpainting. In (a), the inpainting result looks natural, while the (b) shows blurry results. The reason is that the image of (b) lacks of similar texture in the background, which shows the limitation of the patch-based inpainting. Images courtesy of [11].

thesis [14]. This method fills the holes with similar neighbors. Some works propose to fill the region with "pure" repetitive textures of two-dimensional textural patterns with moderate stochasticity. But many natural images contain linear structures and composite textures (multiple textures interacting spatially) [57], which cannot be handled well by this method as shown in Figure 2.5.

Exemplar-based inpainting is an extension of patch-based texture synthesis. This technique not just uses the neighbor patches but finds the best matched ones all around the image. In [11], the authors proposed the improved exemplar-based image inpainting, which combines the advantages of patch-based inpainting and diffusion-based inpainting. They found patch-based filling may be capable of propagating both texture and structure information. And the linear structures abutting the target region is influenced only by the fill order. This exemplar-based inpainting method can get good quality when inpainting the simple textured backgrounds. However, when the inpainting area contains complex textures or structures, especially when no other similar parts exist in another area, this method cannot provide satisfying results. Patch-based inpainting techniques were later accelerated by a randomized patch search algorithm called PatchMatch [2], which allows for real-time high-level image editing of images. The work of [12] demonstrated improved image completion by integrating image gradients into the distance metric between patches. However, these methods depend on low-level features such as the sum of squared differences of patch pixel values, which are not capable of filling complicated structures. Furthermore, they are unable to generate novel objects not found in the source image.

The inpainting methods mentioned above only use the information of a single input image.

Table 2.1: Comparison between traditional inpainting and the inpainting from deep generative models.

	Diffusion-based	Patch-based	Generative inpainting
Information Sources	Single input image	Single input image	**Input image and the external dataset**
Extract features	Based on diffusion formula	Find the most similar patch	**Learned by the generative model**
Fill holes	Propagate the neighbours' value	Copy the patch value	**Directly generate the whole missing parts by network**
Pros	Low computation cost	Low computation cost	**Excellent performance and can restore more different scenes images**
Cons	Can only fill small region, and will lead to blur	Limited by the input image, can not handle complex missing information	**Expensive training cost**

Moreover, these methods require appropriate information to be contained in the input image, like similar pixels, structures, or patches. This assumption is hard to satisfy, if the missing region is large and possibly of arbitrary shape. To overcome this limitation, recent works try to predict the missing pixels using external data. For example, [20] proposed an image completion method using an extensive database of images. They first search for the most similar images to the input in the database and then complete it by cutting the corresponding regions from the matched image. However, this assumes that the database contains an image similar to the input image, which might not be the case. In the same direction, [51] proposed searching images that contained the same scene on the internet to replace the target region. However, the assumption that the same scene is included limits the applicability significantly compared to general approaches. To replace this matching strategy, recently neural networks are used to infer the missing areas of images by learning from plenty of images' structures and textures. The work of [48] proposed to use the denoising autoencoders to reconstruct the images. But the networks like autoencoders can only restore the images that are similar to the images contained in the training dataset since these models can not learn the distribution of the training dataset and generate new examples from the distribution, as it can be performed with deep generative models.

2.3 Inpainting from Deep Generative Models

The state-of-the-art inpainting techniques are based on deep generative models as backbone. Table 2.1 shows the differences of the traditional inpainting and the generative ones.

Before we introduce generative inpainting techniques, we need to have a basic understanding of generative models. Only the models that can generate new samples from the same distribution of the given training data are considered as generative models. A more specific explanation is that the training data has a distribution, which is denoted P_{data}. The generated samples also has a distribution which is denoted as P_{model}. Generative models learn to make P_{model} close to P_{data}. Generative modeling is a density estimation problem. According to the different estimation methods of density, we can divide them into several types, such as generative adversarial networks (GANs)[18], variational autoencoders (VAEs)[27] and autoregressive models. We mainly present VAEs and GANs, since they are the most popular generative models for

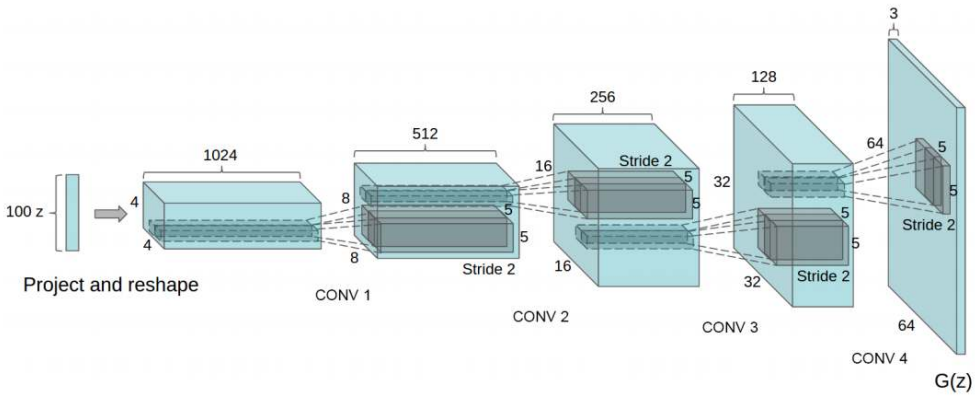


Figure 2.6: The structure of the generator in DCGAN. Image courtesy of [38].

image processing.

2.3.1 Variational Autoencoders

Variational autoencoders are based on autoencoders. An autoencoder takes an input, and reproduce the input as closely as possible. They are composed of two main parts: an encoder that extracts the features (code) from the input, and the decoder exploits the features to reconstruct the original input. The dimension of the code feature is usually smaller than the input because we want features to capture significant factors from the data. Autoencoders cannot be considered as generative models, since they mostly reproduce the information that exist in the training dataset, and cannot generate new samples. To solve this problem, [27] proposed the VAE. Unlike standard autoencoders generating real-valued vectors, VAE's encoder part defines two feature codes: one is the mean code vector (denoted as μ), and the other is the standard deviation code vector (denoted as σ) to obtain a latent vector. The latent vector can produce new samples of similar images.

2.3.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) do not work with any explicit density estimation like VAEs. Instead, it is based on game theory approach to finding Nash equilibrium between two adversarial networks, generator, and discriminator. The generator network model learns to capture the data distribution, and the discriminator model estimates the probability that a sample is from the data distribution rather than model distribution. The two models are trained together in a zero-sum game, until the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples.

Although GANs can generate state-of-the-art samples and is the most popular generative models, their instability becomes a big problem. One of the most stable designs of GANs is the deep convolutional GAN (DCGAN) [38]. Figure 2.6 shows an example of the generator in DCGAN. We can see that all the layers are convolutional layers. This is the most important trick they proposed to improve the stability of GANs: use strided convolutions and fractional-strided convolutions to replace any pooling layers, as well as remove fully connected layers by deeper architectures. Besides this, it uses a batch normalization layer after each convolutional layer except in the output layer for the generator and for the input layer of the discriminator. Although there is no clear theory to support the convergence of these techniques, they provide the inspiration to design our deep generative inpainting model.

Figure 2.7: Facial images generated by a VAE and a DCGAN. The first row samples are from a VAE, while the second row samples are from a DCGAN. Image courtesy of [55].

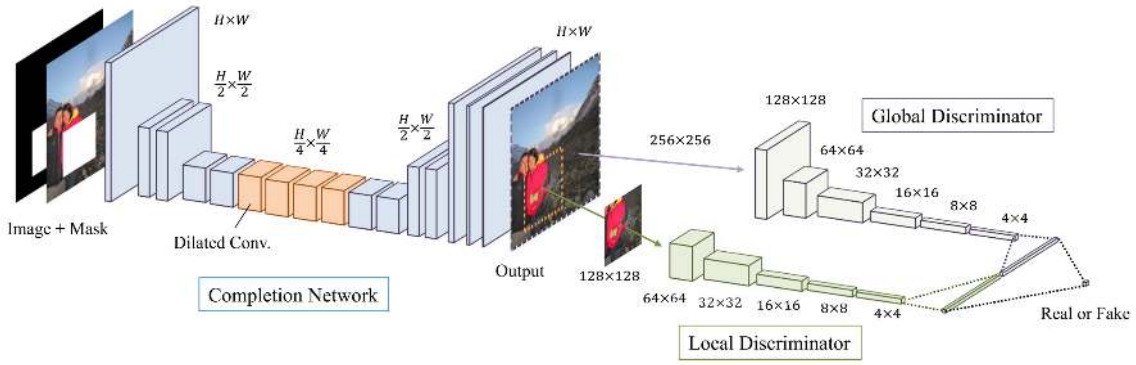


Figure 2.8: The structure of the global and local consistent image completion network. The discriminators are only used to train the completion network. Image courtesy of [21].

State-of-the-art Image Inpainting with Deep Networks

In recent years, there have been many studies on generative inpainting, but most of them are designed for perspective images. The work of [46], they proposed an omnidirectional image inpainting by extracting the region that needs to be restored from the omnidirectional image. Then this region is projected to a local plane (patch rectification) to perform perspective inpainting. However, this method leads to discontinuities at the boundaries of the extracted part.

Several works adopt VAEs to perform generative inpainting [6], [54], because of the more stable performance of VAEs. However, compared to GANs, VAEs tend to generate overly smooth images that are not preferred for inpainting. Figure 2.7 shows some examples generated by a VAE and a DCGAN model. The images from the DCGAN are much sharper than the VAE's. Jointly training VAEs with an adversarial loss prevents the smoothness [28], but may lead to other artifacts.

The model proposed in [21], Globally and Locally Consistent Image Completion (GLCIC) is based on the Context Encoder [34]. The core idea of GLCIC is to use local and global discriminators to discriminate against the inputs jointly. So it can keep globally and locally consistent in the filling regions. GLCIC can handle arbitrary shapes of the missing regions. The structure of the network is shown in Figure 2.8. [55] proposed semantic image inpainting. The core idea is that they first trained DCGAN with real images. And then randomly initializing code and iteratively update it to find the image which contains the closest code to the corrupted images in the dataset. Finally, the missing regions are filled with the found image using Poisson

blending [37]. The main contribution of this work is using features matching improve the inpainting performance. Contextual Attention was proposed by [56], which used a similar matching idea. This network consists of two models. The first model is a simple dilated convolutional network trained with a reconstruction loss. It receives the cropped image and a weight mask and produces a coarse prediction of the missing part of the images. Then the second model has two parallel networks. One is still a dilated convolutional network, while the other one is the contextual attention network. The contextual attention network analyses the generated area in the output image, defining the similarity between each generated and real image pixels. The second model learns how to propagate the real image characteristics to the generated pixels based on this similarity. StructureFlow was proposed by [39], which follows a similar idea of contextual attention. StructureFlow also consists of two models. The first model aims to recover meaningful structures by removing high-frequency textures while retained sharp edges and low-frequency structures. This helps the network to focus on recovering global structures without being disturbed by irrelevant texture information. The second model generates textures based on the output of the first model. Moreover, they used the appearance flow [61] to establish long-term corrections between missing regions and existing regions to help the second model to have better performance on texture generation.

Besides the inpainting for RGB omnidirectional images, the depth ones also need the restoration. In [32], they proposed the vectorial inpainting generative adversarial network (VeIGAN). This network is based on the framework of [56]. Unfortunately, the framework of [56] can not be directly applied to the depth images since the noise will be added in the output, which has a massive impact on the disparity estimation. To solve this problem, they estimate the surface normals of both the generated disparity images and the ground truth ones. And then calculate the l_1 norm of both normal vectors as the vectorial loss to reduce the surface difference. Moreover, they put the coarse prediction result, generated by the first network, with the vector normals to the surfaces.

Beyond images, video inpainting also has passed important milestones. Like single image inpainting, video inpainting is also grounded on model-based methods. They can be divided mainly into two types: object-based methods and patch-based methods. In object-based methods, a pre-processing is required to split the video into foreground objects and background. It is followed by an independent reconstruction and merging step at the end of algorithms. However, the major limitation of these object-based methods is that the synthesized content must be copied from the visible regions. Therefore, these methods are most vulnerable to abrupt appearance changes such as scale variations. In patch-based methods, the patches from known regions are used to fill the missing regions. However, these methods either assume static cameras or constrained camera motions. They are based on a greedy patch-filling process where new errors are inevitably propagated, yielding globally inconsistent outputs [35, 36].

The emergence of deep learning inspire recent works to investigate various deep architectures for video inpainting. One of the state-of-the-art works is the Deep flow-guided video inpainting [53]. This work is designed to restore the optical flow of the consecutive frames, which are the maps that describe the pixels' moving in the video. The optical flow are extracted by the FlowNet model [22], which is a learning-based method for estimating optical flow. The pixels in other frames are propagated to the missing areas (guided by the flow). This method has an excellent performance on video inpainting. Besides, this method is not limited by the diversity of the dataset since what is learned is the objects moving rules in the video. Nevertheless, the structure of this model is too complex and hard to be modified for processing the omnidirectional videos. Not just the flow completion network needs to be changed but also the FlowNet model.

Chapter 3

Omnidirectional Image Processing

In order to design the generative inpainting approach for omnidirectional images, we first need to have a correct understanding of how to represent and process this type of image. These concepts are fundamental since inpainting is done to fill missing information in a plausible way, which strongly depends on understanding the information encoded in the input images. Therefore, knowing how to represent and process panoramic images are prerequisites, such as to extract correct encoded information for inpainting and finally producing realistic inpainting results. This chapter briefly introduces some fundamental image processing concepts, including a short overview of convolutional neural networks. We then introduce omnidirectional images and relevant adapted processing methods to these images.

3.1 Introduction to Image Processing

In this section, we present a few important conventional image processing operators to this project. This is a vast topic ranging from morphological operators to new convolutional neural networks, and therefore a complete introduction is out of the scope of this work. For a complete description, we refer to [15, 17]. We process images with two common goals: *i)* to extract useful information and; *ii)* to enhance the images as in image denoising, super-resolution, to mention a few. Most image processing techniques are defined as "flat" perspective images, often acquired with cameras following the pinhole model. Perspective images are easier to represent, and some operators on them follow the Euclidean space properties. Frequently adopted processing operators on images are:

- Convolutional-based operators: This operator follows the discrete version of the convolution between two functions. The discrete form can be defined as the integral of the two functions' product after one is reversed and shifted. In image processing, the functions are represented by the correlation of the 2D image and a kernel, as shown in Figure 3.1. By using different kernel operators, we obtain a variety of different operators, as the gradient, blurring, image average, among others. For instance, using the Sobel operator (based on gradient) one can extract edge information from the images as depicted in Figure 3.2.

- Erosion and Dilation: They are the most basic morphological operations. Erosion removes pixels on object boundaries in an image, while dilation adds pixels to objects' boundaries. The output of erosion will be the minimum value of all the pixels in the neighborhood, while the dilation is the maximum. For example, in a binary image, the erosion sets a pixel to 0 if any neighboring pixels have the value 0, while the dilation sets a pixel to 1 if any neighboring pixels have the value 1,

- Downsampling and Upsampling: Downsampling decreases the sample rate by an integer factor, which reduces the spatial resolution of the image, while the upsampling does

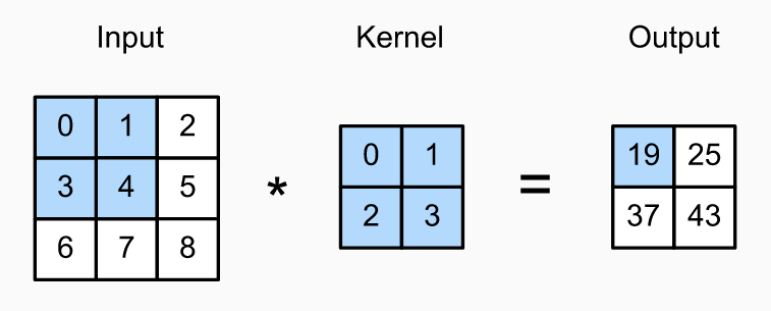


Figure 3.1: A simple image convolution example between (3×3) image and a (2×2) kernel. Image courtesy of [58].

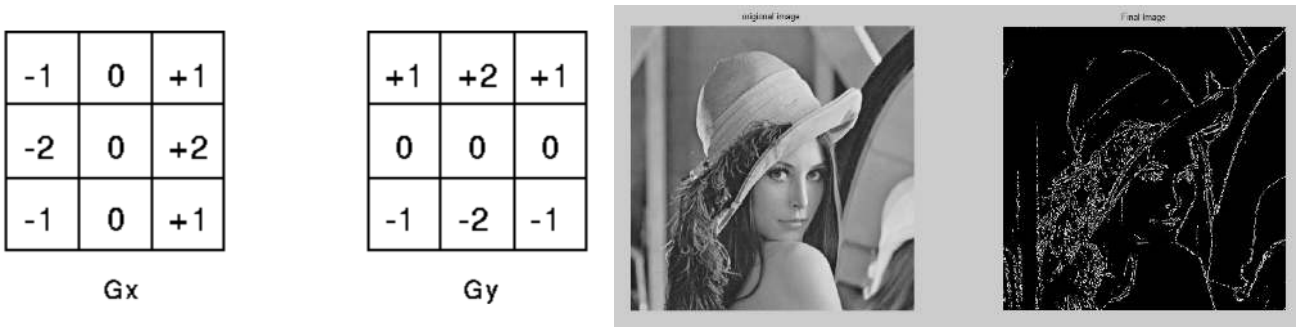


Figure 3.2: Sobel operators. Image courtesy of [40]. $\mathbf{G}_x$ kernel detects vertical edges, while $\mathbf{G}_y$ detects horizontal edges. The result of convolution with these operators with Lenna image is shown at right.

the reverse. There are many methods to do downsampling and upsampling like nearest neighbor resampling, bilinear resampling, and Hermite Resampling. Here we use the nearest neighbor resampling as an example. If we enlarge an image by 2, one pixel will be enlarged to 2×2 area with the same color. If we shrink an image by 2, only 1 pixel over 2×2 pixels is retained in the output image.

- Fourier Transform: It is used to represent the 2D signal encoded in the image by its spectral components. The output of the transformation represents the image in the Fourier or frequency domain, while the input image is the spatial domain. In the Fourier domain, each point represents a particular frequency contained in the spatial domain image. The Fourier Transform is used in a wide range of applications, such as image analysis, image filtering, image reconstruction, and image compression. It will be further addressed later during the omnidirectional processing operators.

Recently, Convolutional Neural Networks (CNN) are frequently used as image processing methods. They combine the aforementioned image operators with artificial neural networks [29]. The main idea of using convolutional layers in CNNs is consecutively extracting features of the input image, in different scales from local to global with convolutions using different kernels, as the CNN architecture shown in Figure 3.3. Among the typical layers of CNNs represent the image operators described previously as:

- Convolutional layers: Using different types of kernels to extract features from the input image.
- Pooling layers: These layers are analog to downsampling operators, reducing the quantity of features by just taking the maximum or mean value of a region of interest on the image.

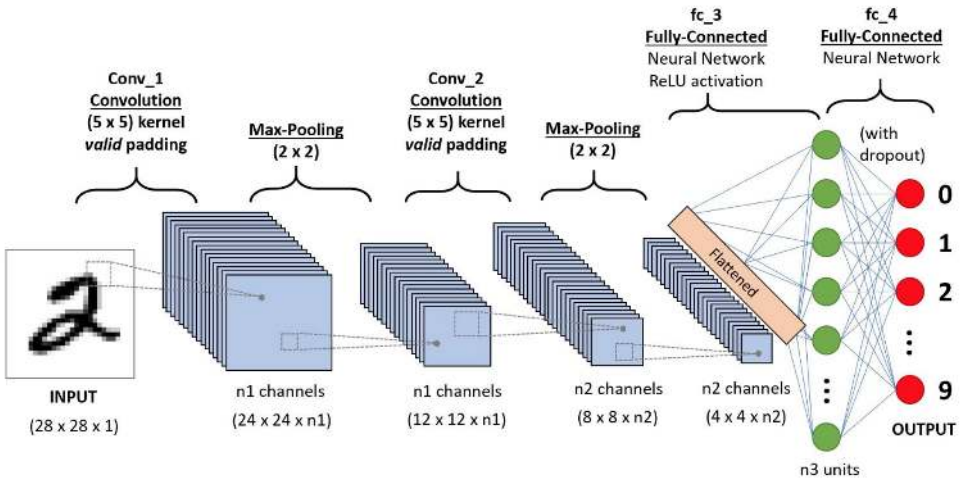


Figure 3.3: A CNN architecture to classify handwritten digits. Image courtesy of [41].

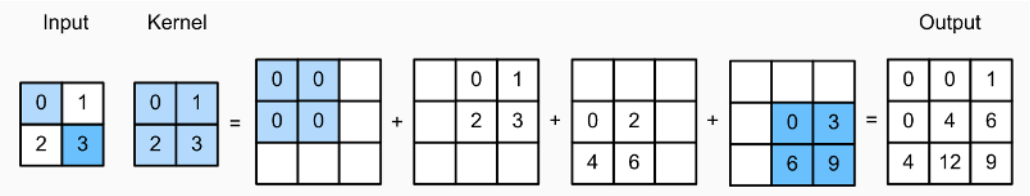


Figure 3.4: Transposed convolution layer with a 2×2 kernel.Both input and output channels are 1, with 0 padding and 1 stride. Image courtesy of [58]

.

- Upsampling layers: These layers are opposite to the polling layers. In order to achieve the goal, we can use the resampling methods mentioned above, or Conv2DTranspose layers, which use the transposed convolution. A simple example of transposed convolution has shown in Figure 3.4.

These operations are the core of neural networks' architectures, and our thesis will build the work on the combination of them. In the final section of this chapter, we discuss some adaptations of these layers that are required to process omnidirectional images adequately.

3.2 Introduction to Omnidirectional Images

Omnidirectional images contain more information than perspective images by increasing the field of view. In order to acquire omnidirectional images, we need special cameras. There are three main types of omnidirectional cameras:

- Dioptric cameras are the combination of shaped lenses slightly bigger than a 180-degree field of view (fisheye lenses).

- Catadioptric cameras combine a standard pinhole camera with a shaped mirror (parabolic, hyperbolic, or elliptical mirror) and provide a 360-degree field of view in the horizontal plane.

- Polydioptric cameras are built from a rig containing multiple cameras (such as perspective or fisheye rings as the Ricoh Theta camera series).

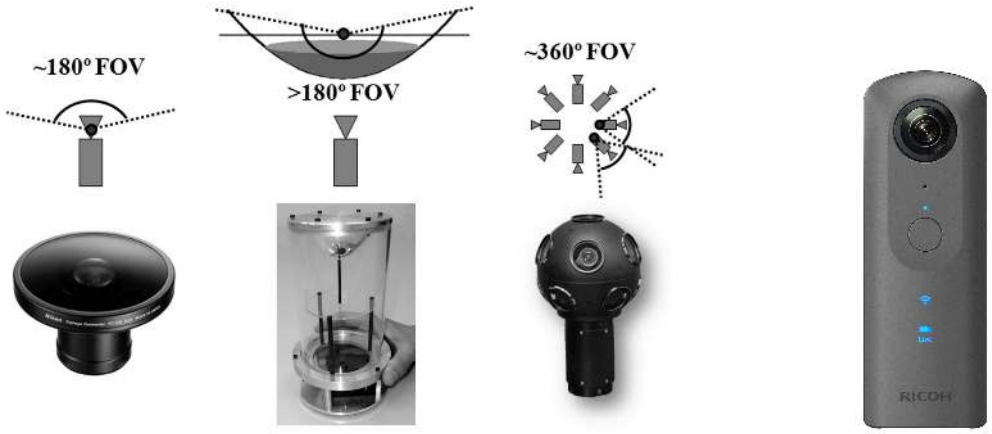


Figure 3.5: The omnidirectional cameras, from the left to right are: dioptric camera(e.g. fish-eye), catadioptric camera, polydioptric camera. Image courtesy of [42]

and the specific polydioptric camera Ricoh Theta that we used to present our results.

Examples of these three cameras are shown in Figure 3.5. In order to manipulate these images, we need to build the right relationship between the 3D points in the world reference frame and the pixels in the image reference frame. We will adopt central camera models in this work, which means that optical rays coming from viewed objects intersect in a single and unique 3D point called projection center or single effective viewpoint. Cameras following this constraint, where every pixel in the sensed image measures the irradiance of the light passing through the viewpoint in one particular direction, allows a unified and elegant representation. The omnidirectional image is mapped to a sphere centered on a single viewpoint (spherical projection).

3.2.1 Camera Projection Models

In order to properly manipulate panoramic images, we need to handle the distortions on these images. So we want to build the camera model and then get the projection model between the 3D points in the world frame and image points. There are different central projection models to represent omnidirectional cameras, here we just present the most representative and widely-used models.

One of the most adopted models is the unified camera model (UCM) [16]. To simplify the computation and calibration, this model has the following two assumptions: *i)* The axis of symmetry of the mirror is perfectly aligned with the optical axis of the camera, and *ii)* The X and Y axis of the camera and mirror are aligned. Under these assumptions, the camera and mirror reference frames differ only by a translation along the Z axis, as shown in Figure 3.6; where C is the mirror reference frame center, C_ϵ is the camera reference frame center, P is a 3D point in the mirror frame, P_s is the point that P project on the unit sphere (centered in C), m is the point that P_s finally project to a normalized image plane, and ϵ is the distance between the mirror frame and the camera frame.

This model allows the projection from 3D points to pixels ($p \in \mathbb{P}^2$) on the omnidirectional image following four steps:

- Projecting the 3D points onto a unit sphere, is done by a spherical normalization as:

$$P_s = \frac{P}{\|P\|} = (x_s, y_s, z_s).$$

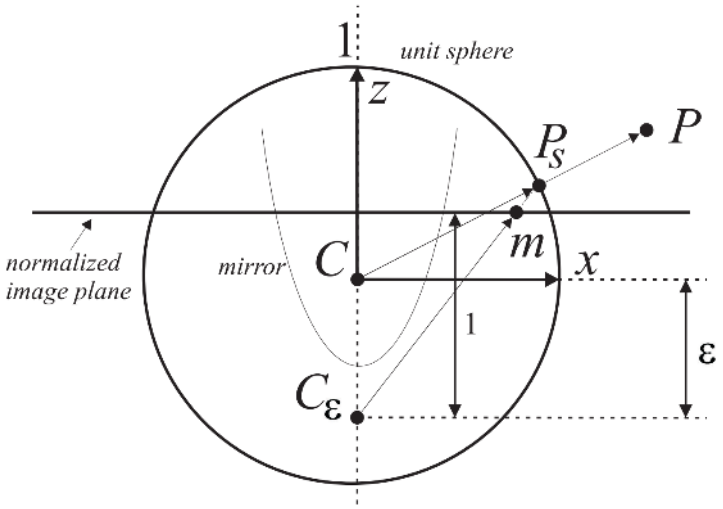


Figure 3.6: The unified camera projection model (UCM) with the detailed notes. Image courtesy of [42].

- The change of the camera frame with translation ϵ along Z. The choice of ϵ is depend on the shape of the mirror (between 0 to 1);

$$P_\epsilon = (x_s, y_s, z_s + \epsilon).$$

- The point P_ϵ is then projected onto the normalized image plane with a perspective transformation (coordinates divide by $z_s + \epsilon$), distant 1 along the positive direction of Z axis from C_ϵ:

$$m = (\frac{x_s}{z_s + \epsilon}, \frac{y_s}{z_s + \epsilon}, 1) = (x_m, y_m, 1).$$

- Finally, the pixel point $p = (u, v, 1)$ is given by the intrinsic-parameter matrix K:

$$p = Km, K = \begin{bmatrix} \alpha_u & \alpha_u cot(\theta) & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The other essential operation is finding the 3D point related to a specific pixel (inverse projection). Due to the spherical normalization, the depth value is lost, so we can just get the 3D point viewing direction with the sphere coordinate constrain $x_s^2 + y_s^2 + z_s^2 = 1$.

$$P_s \propto \begin{bmatrix} x_m \\ y_m \\ 1 - \epsilon \frac{x_m^2 + y_m^2 + 1}{\epsilon + \sqrt{1 + (1 - \epsilon^2)(x_m^2 + y_m^2)}} \end{bmatrix}.$$

This model can represent catadioptric images, but it is also suitable for fisheye images and other images under the central projection assumption, as described in [10]. Due to its generality and importance, this model has been improved over time. For instance, [26] proposed the EUCM, such as the 3D point, do not project into the unit sphere but into a symmetric ellipsoid around the Z axis. Recently, a new extended model named double sphere camera model (DSCM) was presented in [47]. This model considers the 3D point consecutively projected into two unit spheres whose centers are shifted by $\xi = \frac{\alpha}{1-\alpha}$ along Z axis.

The full model has a six parameters, where $\mathbf{i} = \{f_x, f_y, c_x, c_y, \xi, \alpha\}$. The projection function (from 3D ($\mathbf{x} = \{x, y, z\}$) point to 2D ($\mathbf{u} = \{u, v\}$) point) is as follows:

$$\pi(\mathbf{x}, \mathbf{i}) = \begin{bmatrix} f_x \frac{x}{\alpha d_2 + (1-\alpha)(\xi d_1 + z)} \\ f_y \frac{y}{\alpha d_2 + (1-\alpha)(\xi d_1 + z)} \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \tag{3.1}$$

where $d_1 = \sqrt{x^2 + y^2 + z^2}$, $d_2 = \sqrt{x^2 + y^2 + (\xi d_1 + z)^2}$.A set of 3D points that results in valid projection is expressed as follows:

$$\Omega = \{\mathbf{x} \in \mathbb{R}^3 \mid z > -w_2 d_1\}. \tag{3.2}$$

where the w_2 is:

$$w_2 = \frac{w_1 + \xi}{\sqrt{2w_1\xi + \xi^2 + 1}}, and\ w_1 = \begin{cases} \frac{\alpha}{1-\alpha}, if \alpha \leq 0.5 \\ \frac{1-\alpha}{\alpha}, if \alpha > 0.5 \end{cases} \tag{3.3}$$

The inverse projection function is:

$$\pi(\mathbf{u}, \mathbf{i})^{-1} = \frac{m_z\xi + \sqrt{{m_z}^2 + (1-\xi^2)\rho^2}}{{m_z}^2 + \rho^2} \begin{bmatrix} mx \\ m_y \\ m_z \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \xi \end{bmatrix}. \tag{3.4}$$

where $m_x = \frac{u - c_x}{f_x}$, $m_y = \frac{v - c_y}{f_y}$, $\rho^2 = {m_x}^2 + {m_y}^2$, $m_z = \frac{1 - \alpha^2\rho^2}{\alpha\sqrt{1-(2\alpha-1)\rho^2} + 1 - \alpha}$.

And the resulting domain is:

$$\Theta = \begin{cases} \mathbb{R}^2, if\ \alpha \leq 0.5 \\ \{\mathbf{u} \in \mathbb{R}^2 \mid r^2 \leq \frac{1}{2\alpha - 1}\}, if\ \alpha > 0.5 \end{cases} \tag{3.5}$$

The DSCM better fits cameras with fisheye lenses, has a closed-form inverse, and does not require computationally expensive trigonometric operations. Another point that has been improved overtime is the computation of the projection and inverse projection operations. In this sense [42] computes the inverse projection from the image point to a 3D point by using Taylor polynomial approximation:

$$P_s \propto \begin{bmatrix} x_m \\ y_m \\ f(\rho) = a_0 + a_2\rho^2 + \cdots + a_N\rho^N \end{bmatrix}, and\ \rho = \sqrt{x_m^2 + y_m^2}.$$

The author proved that coefficient a_1 is always zero. So the formula not include this item. The coefficients value of the polynomial $(a_0, a_2, \cdots a_N)$ and polynomial degree N are found through a calibration process. This model also called Taylor model. Based on this inverse projection model, we can build the projection model as well. It treats the imaging system as a unique compact system. The relationship between the 3D points and pixel points is shown in Figure 3.7. In the figure, $\boldsymbol{x}$ is the scene point, and $\boldsymbol{u''}$ (equal to the point m mentioned above) is the perfect project point on image (assume that the physical system is perfect). So we will have the follow relationship:

$$\lambda * g(\boldsymbol{u''}) = T\boldsymbol{x}, \tag{3.6}$$

where λ is the depth scale, function $T \in \mathbf{R}^{3\times4}$ is the transfer matrix that project $\boldsymbol{x}$ from world frame to camera frame and $g(\boldsymbol{u''}) = (u'', v'', f(\rho))^T$. If the physical system is perfect, then we just need to calibrate the coefficients of function $f(\rho)$ and the extrinsic parameters of P. However, the physical system is not perfect, which will lead the points' transform. So the

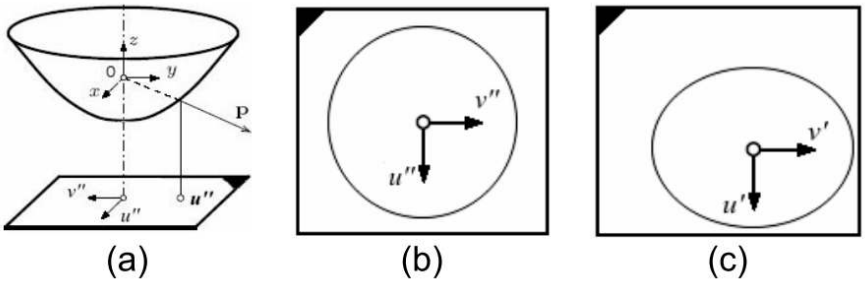


Figure 3.7: Taylor camera model. (a) Coordinate system in the catadioptric case. (b) The perfect projection plane. (c) The actual camera image plane, expressed in pixel coordinates. (b) and (c) are related by an affine transformation. Image courtesy of [43].

actual point position $\boldsymbol{u'}$ have the relationship as $\boldsymbol{u''} = \boldsymbol{A}\boldsymbol{u'} + \boldsymbol{t}$. $\boldsymbol{A}$ is a 2 by 2 affine matrix for distortion, and $\boldsymbol{t}$ is the translation between image center and the omnidirectional image center.

From the world to camera, we can get the point direction directly from the x, y coordinates, but we also need to know the distance from the center to the 2D points. So we can also use a polynomial to reconstruct the distance. The polynomial function for world to camera is:

$$f_{inv}(\theta) = \rho = a'_0 + a'_1\theta + a'_2\theta^2 + \cdots + a'_N\theta^N, \theta = atan(\frac{z}{\rho}).$$

After getting the parameters of the two projection polynomial function, we can easily approximate the point position.

3.2.2 Camera Calibration

The camera model parameters can often not be directly estimated, but by using the images that contain a calibration pattern. We need to estimate two types of parameters: *i)* transformation between the camera frame and the world frame, which called extrinsic parameters; *ii)* and camera model intrinsic parameters. We denote G as the total project function, which can project the 3D points into the image. And we hope the project points the closer to the actual image points the better. So we can formulate the problem as:

$$\boldsymbol{V} = \arg\min \frac{1}{2}\sum_{i=1}^{m}[G(\boldsymbol{V}, \boldsymbol{g}_i) - \boldsymbol{e}_i]^2 \tag{3.7}$$

where $\boldsymbol{V}$ is the total parameters, $\boldsymbol{e}_i$ is the pixel coordinate and $\boldsymbol{g}_i$ is the 3D coordinate. This formula aims to find a set of parameters that can minimize the distance between the pixel points in the camera frame and the points project from the 3D world frame. We can calibrate any camera model just by replacing the function G and parameters V. Different algorithms can be used to minimize this function like gradient descent, simulated annealing, and Levenberg-Marquardt. In [33], they propose a G function as:

$$G = P \circ D \circ H \circ W,$$

where W, H, D, P correspond to extrinsic transformation, transformation between camera frame and mirror frame, distortion function, generalised camera projection. This method is based on UCM. But it can also adjust to EUCM and DSCM showed in the previous section.

Because the Taylor model uses approximation but not geometry in computing the points, it is G function does not have an exact formula, and the way to solve the calibration is also more complex. More details about the calibration can be seen in [43] and including a toolbox [44].

Figure 3.8: The visual results of the calibration of UCM. These images come from the Ricoh Theta camera's back camera. Green points mean the detected corners, the red points are the 3D points project on the image based on the parameters from calibration, the red boundary is the project plane.

3.2.3 Calibration Results

Results of UCM Calibration

We used 30 images for UCM calibration, which come from the Ricoh Theta camera's back camera. The final visual results are shown in Figure 3.8. To evaluate the calibrated model's accuracy, we can compute the mean reproject error (MRE, here we define the error as the euclidean distance between the detect points and the reproject points) as:

$$MRE = \frac{\sum \sqrt{(x - x_d)^2 + (y - y_d)^2}}{N}, \tag{3.8}$$

where x, y are the project points coordinates, x_d, y_d are the detect points, and N is the number of points. The final error is 1.315 pixels, and the standard deviation is 0.5794 pixels. This error is small enough since we need to compare it with the magnitude of pixels number in the general image. From visual inspection of the reprojection result in Figure 3.8, we almost cannot find the perceptual difference by the human eye. All these results prove that under the calibration build in the UCM, we can build the right relationship of the pixel points and the 3D points.

Results of the Taylor Model Calibration

In order to compare the calibration results, we use the same images set as the above model. After testing with different degrees of polynomials, we finally choose the degree as 5 to the projection, and the corresponding degree of the inverse polynomials is 15. To know the project accuracy of this model, we show one of reproject results in Figure 3.9. The MRE is 0.473463 pixels, and the standard deviation is 0.317803 pixels. We can see that most errors are less than one pixel. In comparison with the UCM, this model shows better performance on the projection. Although this model has a more sophisticated calibration procedure, it has a more concise projection model than the UCM. This model is also efficient computationally so that we will choose this projection model for the following image processing.

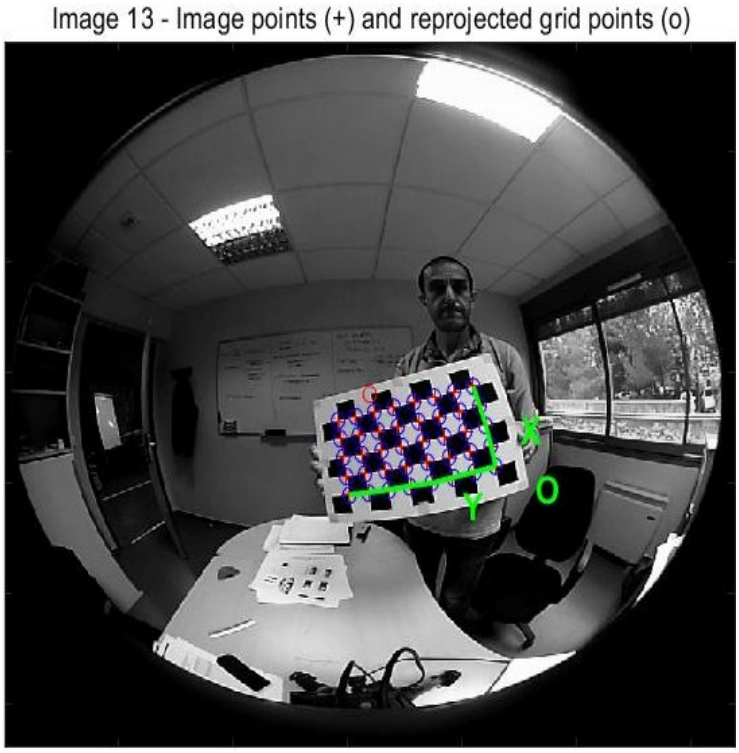


Figure 3.9: The reprojected points and pixels on the checkerboards.

3.3 Omnidirectional Image Mapping to the Unit Sphere

All the following processing methods will require the relationship between the omnidirectional image and unit sphere surface, so we need to build the mapping function between the two presentation ways. And a useful representation of unit sphere surface is the equirectangular plane (in θ, ϕ coordinate). The relationship between these two presentation will be a little different depending on the definition of the coordinates. Generally we will use the follow relationship:

$$P_s = (cos(\phi)sin(\theta), sin(\phi), cos(\phi)cos(\theta), \theta \in [0, 2\pi], \phi \in [0, \pi]). \tag{3.9}$$

Actually there are forward wrapping and backward wrapping, as the forward one will lead to some areas cannot find corresponding value. So we decide to use the backward wrapping. Mapping function from the omnidirectional image to (θ, ϕ) plane can be divided into following steps:

- First we do the discretization of (θ, ϕ) to get the relationship of the pixels' position and their corresponding degree: $\theta_k = \frac{2k\pi}{N}, \phi_k = \frac{k\pi}{N}$.
- And then we project the (θ_k, ϕ_k) to the unit sphere surface by the coordinate relationship mentioned above: $p_u = (cos(\phi_k)sin(\theta_k),$
  $sin(\phi_k)sin(\theta_k), cos(\theta_k))$.
- Finally we can project the spherical points to omnidirectional image by: $p = K * P_s$, where K is the intrinsic matrix which get from calibration.

We can get the corresponding coordinates from the above mapping, and then just put the corresponding color value to the (θ, ϕ) plane. The mapping function from (θ, ϕ) plane to the omnidirectional image can be summarized as follows:

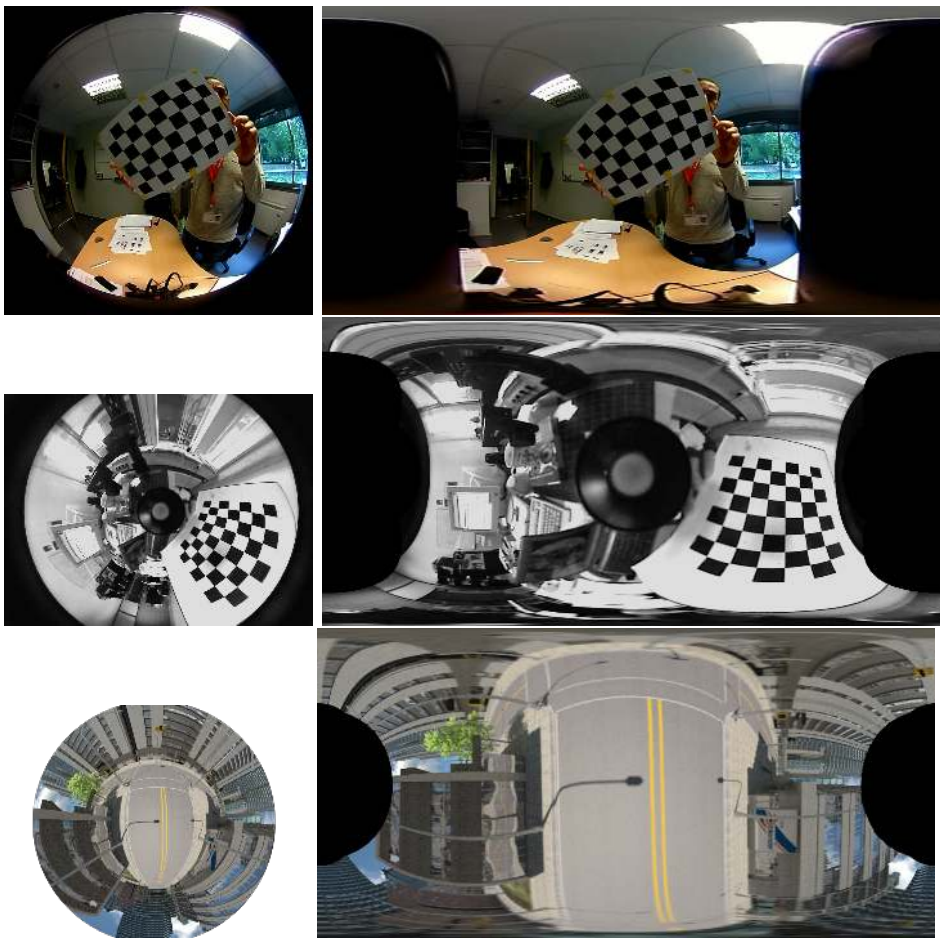


Figure 3.10: The (θ, ϕ) equirectangular image project from omnidirectional image. The first column images are the original omnidirectional images (the first one from fisheye camera, and other two from catadioptric camera). The second column images represent the equirectangular projection.

- We project the pixel coordinate (u, v) to the unit sphere surface using Taylor model with the polynomial of degree 5.
- And then we project the (x, y, z) to the (θ, ϕ) plane by $\phi = \arccos(y), \theta = \arctan(x, z)$.
- Finally, we do the discretization to get the pixel coordinates: $\theta_k = (0.5 + \theta/(2\pi)) * width, \phi_k = (0.5 + \phi/\pi) * h$

3.3.1 Results of the Equirectangular Projection

In order to test our mapping function, we apply it to different omnidirectional image types. These sequences include fisheye images were shown in the previous section, while the other two catadioptric images from [44] and [60]. Some mapping results are shown in Figure 3.10. From the images, we can see limited by the field of view, when the omnidirectional images can not entirely cover all the surface of the unit sphere. And we can see that the catadioptric image has a larger field of view than the fisheye image.

Some examples of the final mapping result are shown in Figure 3.11. The back project images are almost the same as the original images. We compute several similarity metrics to compare the similarity of the back project images and the original images. In order to know the sensitivity of the different metrics, we also use incorrect parameters to generate some images, which are shown in Figure 3.12.

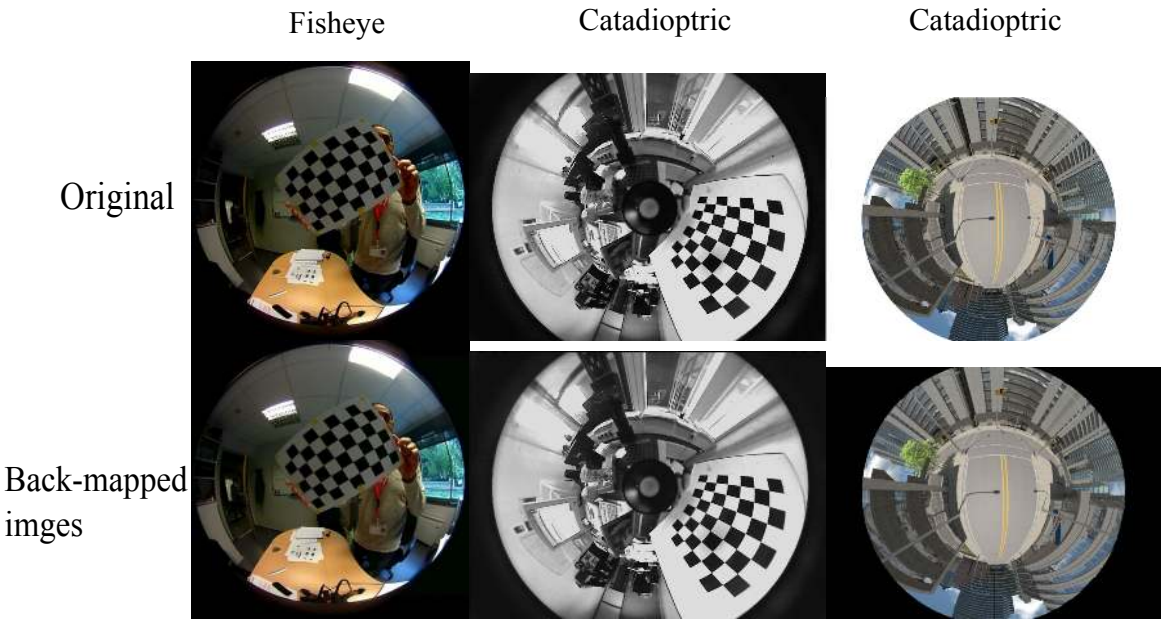


Figure 3.11: Back-projected results. The first row images are the original images. The second-row images are the back mapping images using the mapping operations.

The most common evaluation metrics are mean square error (MSE), structural similarity index metric (SSIM), and image histogram . MSE is an index value used to calculate the similarity of two matrices. The smaller the value is, the more similar the two matrices are. It requires the two matrices have the same shape. Its calculation formula is (for the version of the images):

$$MSE = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}[I(i,j) - K(i,j)]^2, \tag{3.10}$$

where m, n are the rows and cols of the image, and I, K are the two images.

SSIM can better reflect the similarity of two pictures. The range of this indicator is $[-1, 1]$. When $SSIM = -1$, it means that the two pictures are completely dissimilar. When $SSIM = 1$, it means two pictures. very similar. The closer to 1, the more similar the two pictures are. This index is computed as follows:

$$SSIM = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{({\mu_x}^2 + {\mu_y}^2 + c_1)({\sigma_x}^2 + {\sigma_y}^2) + c_2}, \tag{3.11}$$

where x, y are the two compared images, μ_x, μ_y are the average x, y, ${\sigma_x}^2, {\sigma_y}^2$ are the covariance of x, y, c_1, c_2 are two variables to stabilize the division with weak denominator.

The main idea of image Hash is that each image is reduced down to a small hash code or 'fingerprint' by identifying salient features in the original image and hashing a compact representation of those features (rather than hashing the image data directly). Finally, by counting the number of bit positions that are different (Hamming distance). The Hash value closer to 1 means more similarity. There are three types of Hash: *i)* Average Hash (aHash), for each of the pixels output one if the pixel is bigger or equal to the average and 0 otherwise; *ii)* Perceptive Hash (pHash), which does the same as aHash, but first, it does a Discrete Cosine Transformation and works in the frequency domain; *iii)* Derivative Hash (dHash), which calculates the difference for each of the pixels and compares the difference with the average differences. Concerning the computational aspect, dHash is as fast as aHash and very few false positives. Therefore we will dHash for the evaluation.

The results of computing the similarity with different metrics are shown in Table 3.1. We can find that MSE is very sensitive to the change. Even a slight change will produce a strong

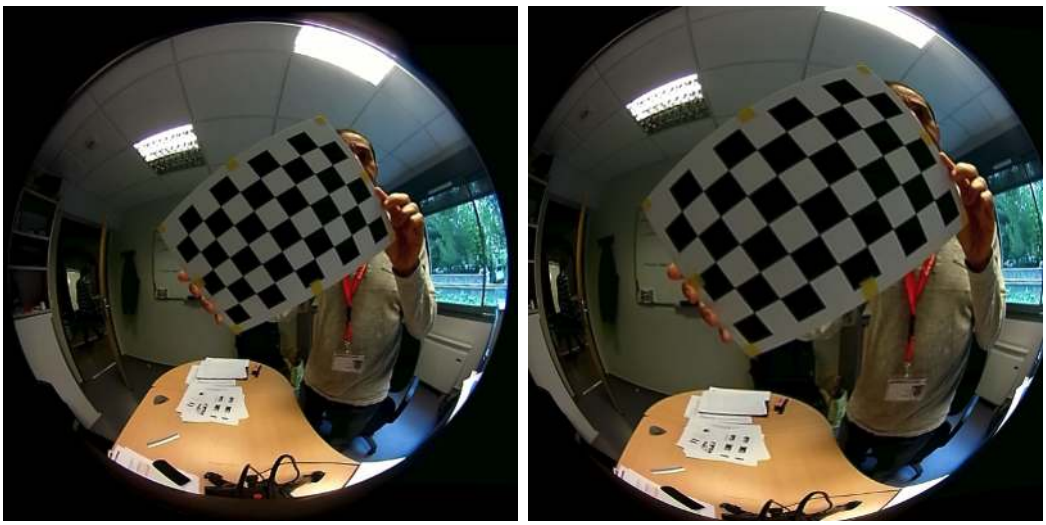


Figure 3.12: Examples of back-projected images with wrong parameters. The left view is slightly changed and right one is huge changed.

Table 3.1: Different similarity metrics applied on the images of Figure 3.11

Back project images	MSE	SSIM	dHash
Fisheye	333.35	0.79	0.922
Catadioptric 1	503.10	0.79	0.875
Catadioptric 2	144.47	0.82	0.902
Minor calibration error (fisheye)	1426.46	0.71	0.859
Strong calibration error (fisheye)	5109.60	0.40	0.648

variation. The SSIM and dHash, which can still recognize a similar image with slightly changed, means these two metrics can perform better on the similarity of the whole structure.

Image histogram, which counted the number of pixels with different values, can present the distribution of the images. The result is shown in Figure 3.13. Although the image histogram is not entirely the same, they have the same trend. The histogram similarities from 0 to 1, the bigger means, the more similar. But the results also show that histograms cannot well recognize the changes, since the larger incorrect image has higher similarity in the histogram. So we should combine with other metrics to compute the similarity.

Combining the results of MSE, SSIM, dHash, and image histograms, we can see that although the back project image seems the same as the original one from the human eye, actually the information is changed or even lost after these complex transform calibrations. However, these results also show that the back project images still keep the primary information of the original images. Thus we can say that we successfully build the mapping function for processing the omnidirectional image.

3.4 Omnidirectional Image Processing Methods

Even with the representation of omnidirectional images described in the previous section, we can not directly use the standard image processing methods on them. This is due to most image processing methods to adopt an isometric Euclidean space for all pixels. This is valid for perspective flat images, where pixel regions have physically the same influence on neighbors at the same distance. But in omnidirectional images, the influence of a pixel on its neighbors

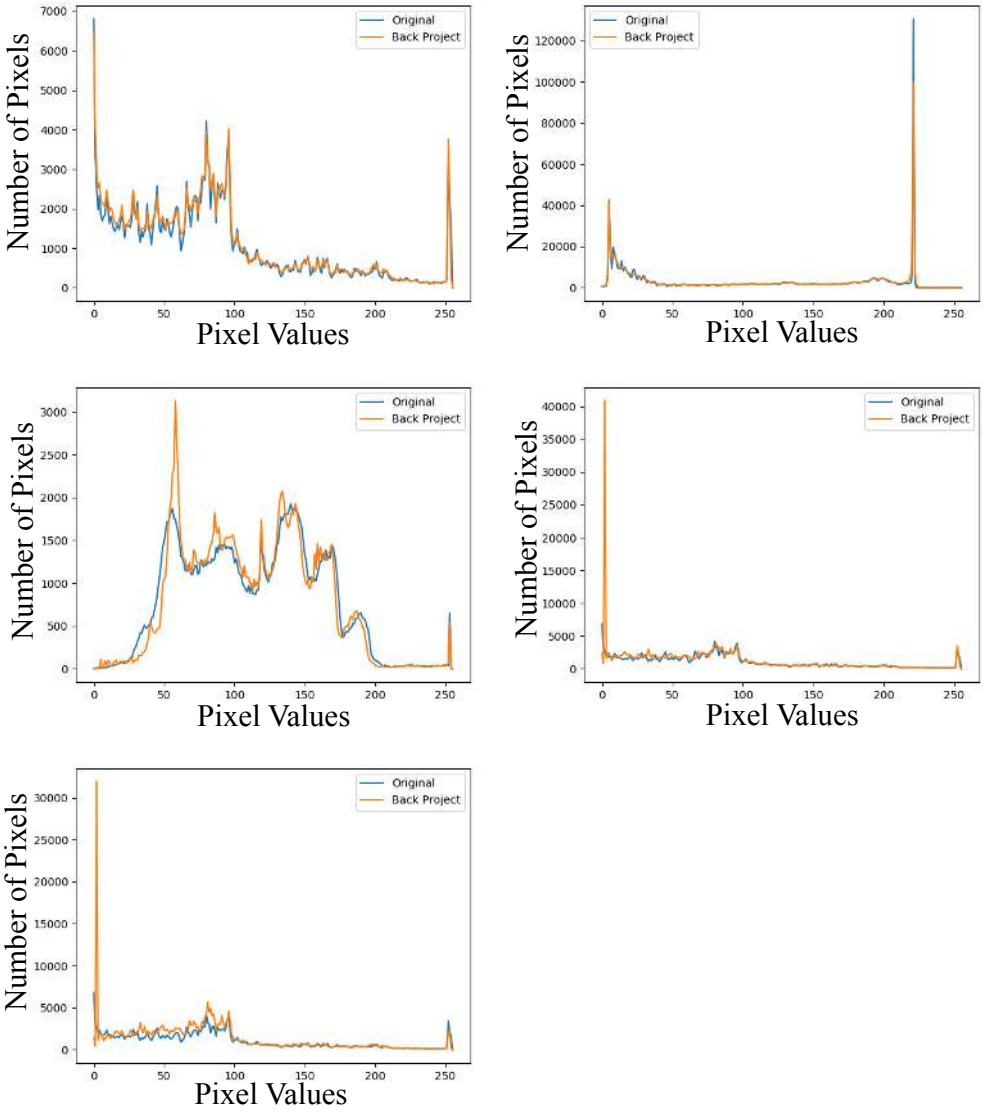


Figure 3.13: Histogram of the original images and the back-projected ones. From top to the bottom correspond to the same order images in Figure 3.11 and Figure 3.12. And the similarities of the histogram from top to bottom and from left to right are 0.985, 0.987, 0.957, 0.392 and 0.503.

strongly depends on its position in the image because of the distortions. In often words, the pixel distribution is not spatially invariant in the image.

To overcome these problems, we need to redefine conventional image processing methods to adapt to omnidirectional images. There are two main possible directions in order to process omnidirectional images. One is by redefining the pixel neighboring relations, which means choosing the neighbors that will have the same physically influence on the pixels and then projecting the omnidirectional image to a unit sphere where we could consider the same distance neighbors' physically influence will not be changed when the pixels' position changes. The second one is using spectral analysis. But we need the redefinition the processing methods that can process the pixels on the sphere. This section introduces the main ideas of these directions and important related work.

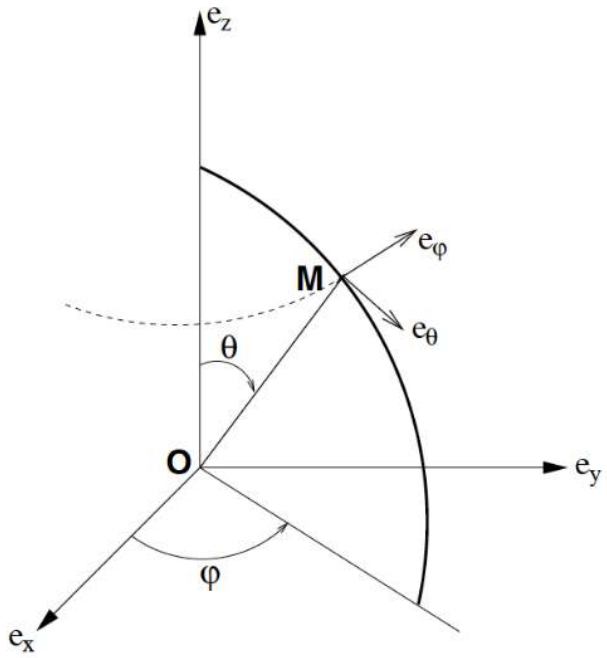


Figure 3.14: Spherical coordinates and local basis for defining pixel neighbourhood. Image courtesy of [13].

3.4.1 Redefining Pixel Neighboring and Projection Approaches

The work in [13] propose to redefine the pixel neighbors with the geodesic metric. So the change of pixel neighborhood caused by the distortions can be handled. Let denote P the projection from a central catadioptric image to the unitary sphere:

$$P : \mathbf{x}(x, y) \rightarrow \mathbf{x_s}(\theta, \phi), \mathbb{R}^2 \rightarrow S^2, \tag{3.12}$$

where

$$\mathbf{x_s} = \begin{bmatrix} \cos\phi \sin\theta \\ \sin\phi \sin\theta \\ \cos\theta \end{bmatrix}. \tag{3.13}$$

The distance of two points on the sphere can be defined as:

$$d_{S^2}(\mathbf{x_s}, \mathbf{y_s}) = \arccos\left(\mathbf{x_s}^T \mathbf{y_s}\right). \tag{3.14}$$

We can then define a continuous neighborhood using this geodesics:

$$V_r(\mathbf{x}) = \{\mathbf{y_s} \in S^2, d_{S^2}(\mathbf{x_s}, \mathbf{y_s}) \leq r \mid P(\mathbf{x}) = \mathbf{x_s}\}. \tag{3.15}$$

In the discrete version, we need to consider a tangent plane at $\mathbf{x_s}$, as shown in Figure 3.14 exemplifying this transformation.

The tangent plane is represented by the unit vectors:

$$\pi = \mathbf{x_s} + a\boldsymbol{e}_\theta + b\boldsymbol{e}_\phi : \tag{3.16}$$

where the unit vectors can be defined as:

$$\begin{aligned} \boldsymbol{e}_\theta &= \frac{\partial \overrightarrow{OM}}{\partial \theta} = \begin{bmatrix} \cos\phi\cos\theta \\ \sin\phi\cos\theta \\ -\sin\theta \end{bmatrix}, \\ \boldsymbol{e}_\phi &= \frac{1}{\sin\theta}\frac{\partial \overrightarrow{OM}}{\partial \theta} = \begin{bmatrix} -\sin\phi \\ \cos\phi \\ 0 \end{bmatrix}. \end{aligned} \tag{3.17}$$

In order to obtain $(2N+1)^2$ points (per row and column contains $(N-(-N)+1)$ points) from π:

$$\mathbf{x_s} + \tan(nr)e_\theta + \tan(pr)e_\phi, -N \leq n, p \leq N. \tag{3.18}$$

Then we project these points from the tangent plane to the unit sphere:

$$V_s^N(\mathbf{x}) = \{\mathbf{x_s}(n,p) = \frac{\mathbf{x_s} + \tan(nr)e_\theta + \tan(pr)e_\phi}{\left\|\mathbf{x_s} + \tan(nr)e_\theta + \tan(pr)e_\phi\right\|}, -N \leq n, p \leq N, P(\mathbf{x}) = \mathbf{x_s}\} \tag{3.19}$$

Finally, we can project the geodesic neighborhood to the original image plane. And we can apply the convolution product and other methods with little adaptations adopting the geodesic metric based neighbor.

This approach allows us to directly adapt the classical perspective image processing while considering the distortions. Contrary to methods based on spherical harmonic analysis (detailed in the next subsection), this method process points only in the image plane, which means a simpler and faster computation. [25] also propose to project the omnidirectional image onto a virtual cylinder (locally planar). However, this model is not appropriate to spherical images, which requires anisotropic operators on the spherical image and then processing changes as the pixel position changes in the omnidirectional image. Here we just show how to choose the neighbor in the unit sphere surface with a tangent plane:

- Generate the basic vectors of the tangent plane:

  $$t_x = |v \times p_u|, t_y = |p_u \times t_x|,$$

  where p_u is the center spherical coordinates, and $v = (0, 1, 0)$.

- And then just to get the neighbors' coordinates by the linear combination the basis vectors.

Finally, we can use the back mapping function described in the previous section to project the neighbors into the equirectangular plane. The neighbors' project in the equirectangular plane is shown in Figure 3.15. Limited by space, we can only show several examples of the new neighbors. We can see that the neighbors' position will change as the center changed. When it gets closer to $\phi = 0, \pi$, the neighbors will have more dispersion, while pixels close to the center become more compact.

3.5 Spherical Image Processing with Spectral Analysis

By projecting the redefined distortion-aware neighbors back to the equirectangular plane, we can directly apply the perspective image processing methods on it. However, every time we need to compute the corresponding positions on the unit sphere surface if we change the size of the kernel. So why not directly process the image in the spherical domain. In this way, we need

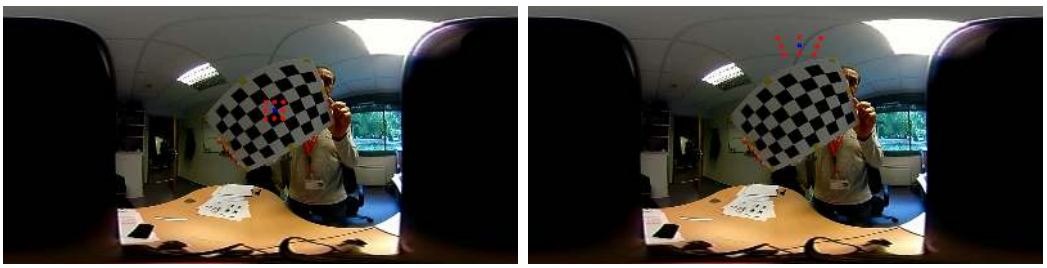


Figure 3.15: Spherical neighbours project on the equirectangular plane. The new neighbors (red points) of the point (blue point) in the equirectangular plane.

to model the distortion of the kernel. The second strategy consists of turning the signal into the frequency domain by Fourier transform. Although this method seems more complex than the former one, it has several nice properties. In the next several subsections, we will introduce how we can represent the image in the frequency domain, the properties of this representation, and finally how we can define the spectral convolution on them.

3.5.1 Orthogonal Basis Functions

The Fourier transform encodes a signal into a linear combination of periodic orthogonal functions. This basis functions actually are some simple signals like functions $sin(nx)$ and $cos(nx)$. The main properties of these functions are that they can be scaled and linearly combined to approximate the original function. Like the Euclidean geometry, Euclidean vectors can be multiplied by standard base points to obtain the projection length. We denote the original function as $f(x)$ and the basis function as $B_i(x)$. We can get the weight of the basis functions by projecting $f(x)$ onto $B_i(x)$ by integrating the product $f(x)B_i(x)$ over the full domain.

The basis function is a family of functions with orthogonal polynomials. The orthogonality here is a generalized orthogonality in function space and vector space. The inner product of two different orthogonal polynomials in a given domain is 0, while the same orthogonal polynomials is a constant value:

$$\int F_m(x)F_n(x)dx = \begin{cases} 0 & for\ n \neq m \\ c & for\ n = m \end{cases} \tag{3.20}$$

This property allows the basis functions to not "overlap" each other's influence while still occupying the same space. Furthermore, it will make the rotation of the function easier, which would be represented in the next section. In the families of the orthogonal polynomials, we are especially interested in the Legendre polynomials, in particular the Associated Legendre Polynomials (ALP). ALP is the solution of Legendre Differential Equation equivalent to solving the three-dimensional Laplace equation in spherical coordinates:

$$P_l^m(x) = \frac{(-1)^m}{2^l l!}(1-x^2)^{m/2}\frac{d^{l+m}}{dx^{l+m}}(x^2-1)^l. \tag{3.21}$$

ALP is defined over the range [–1,1] and returns real numbers. There are two arguments l and m, which are the band index and degree because the functions are broken into bands. So l indicates the band from 0 to any positive integer value. And m indicates the polynomial element index of the current band, which takes an integer value in the range $[0, l]$.

An important property of these functions: inside a band, the polynomials are orthogonal with the same constant value, while between different bands, they are orthogonal with a different constant.

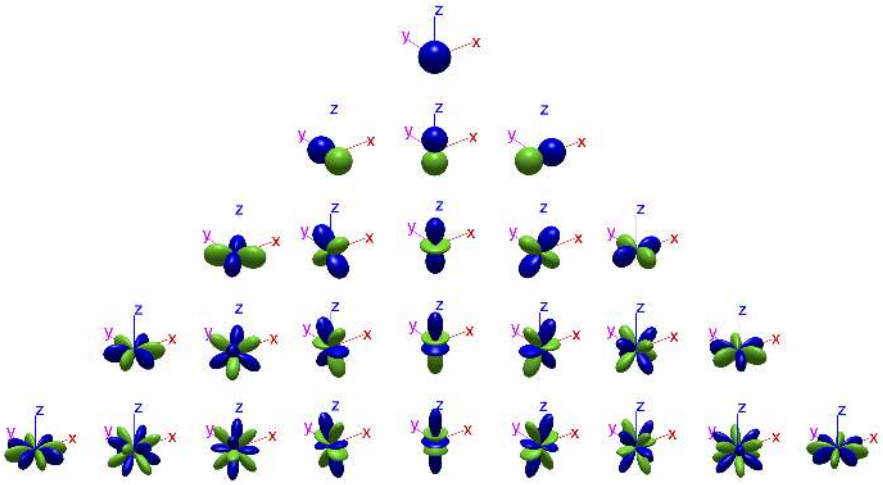


Figure 3.16: Visualization of spherical harmonics. The first five bands of spherical harmonics. The blue part means the real part of spherical harmonics is positive, while the green one represent the negative part.

Generally, we not directly solve the $P_l^m(x)$, since the calculation complexity is very high. In high-order cases, we will use recursive iterations to solve the problem:

$$(l-m)P_l^m = x(2l-1)P_{l-1}^m - (l+m-1)P_{l-2}^m \tag{3.22}$$

$$P_m^m = (-1)^m(2m-1)!!(1-x^2)^{m/2} \tag{3.23}$$

$$P_{m+1}^m = x(2m+1)P_m^m \tag{3.24}$$

Alternate using these three formulas we can generate all the elements of the polynomials.

3.5.2 Spherical Harmonics

As we mentioned above the ALP are equivalent to the solution of the three-dimensional Laplace equation in spherical coordinates which we also call spherical harmonics (SH). So we can present the SH based on the ALP as [3]:

$$Y_{lm} = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}P_l^m(cos(\theta))e^{im\phi}, \tag{3.25}$$

This equation is the general form defined in the complex field. The coefficient $\sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}$ is a scaling factor used to normalize this basis. Note that although the degree parameter m of ALP is a non-negative integer, the parameter m of the spherical harmonic function can be taken to be the negative number in the range $[-l, l]$. The visualization of the first five bands of SH is shown in Figure 3.16.

Finally, the image intensity function is defined on a sphere $\boldsymbol{I} \in L^2(S^2)$ can be expanded into a double generalized Fourier series with the spherical harmonics:

$$\boldsymbol{I} = \sum_{l \in N}\sum_{|m| \leq l} \hat{\boldsymbol{I}}_{lm} Y_{lm} \tag{3.26}$$

where $\hat{I}_{lm}$ is the coefficients of the spherical Fourier transform of the signal I:

$$\hat{I}_{lm} = \int_{S^2} I(\eta) Y_{lm}^*(\eta) d\eta. \tag{3.27}$$

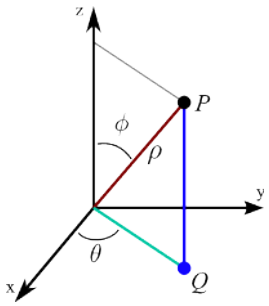


Figure 3.17: The coordinate system of spherical image. The coordinate of P is $(x, y, z) = (cos(\theta)sin(\phi), sin(\theta)sin(\phi), cos(\phi))$

We should also clearly point out the coordinates system definition. As shown in the Figure 3.17, the position we look at the sphere is that the x axis point to us. In actual operation, the sphere integral is generally not easy to directly obtain from the analytical formula. We can use Monte Carlo integral estimator is often used to approximate:

$$\int_S f ds \approx \frac{1}{N} \sum_{j=1}^{N} f(x_j) w(x_j), \tag{3.28}$$

where the weight $w(x_j) = \frac{1}{p(x_j)}$. So for uniform spherical sampling, the weight is:

$$w(x_j) = \frac{1}{p_(x_j)} = \frac{1}{(\frac{1}{S_{sphere}})} = 4\pi. \tag{3.29}$$

So the numerical estimation expression of the SH coefficient is:

$$\begin{aligned} \hat{I}_i &= \int_S I(\mathbf{v}) Y_i(\mathbf{v}) ds \\ &\approx \frac{1}{N} \sum_{j=1}^{N} I(x_j) Y_i(x_j) \cdot 4\pi, \\ &= \frac{4\pi}{N} \sum_{j=1}^{N} I(x_j) Y_i(x_j) \end{aligned} \tag{3.30}$$

where the coefficient is reshaped into one-dimensional vector. As the band number l increase, the reconstruct function will also get closer to the original one. However, this value cannot be increased to infinite in practical applications. We need to choose a certain value of l which called band limited approximation. In this context, high-frequency signals greater than a certain threshold are removed due to the limitation of the frequency-domain signal bandwidth.

We reconstruct the spherical image with different bandwidth to evaluate the effect of spherical harmonics. The results for an equirectangular image are shown in Figure 3.18. We can see that as the bandwidth increases, the image also becomes closer to the original image, and more information is kept. However, we can find that the images contain more corners and the thin lines are hard to rebuild. Because in the frequency domain, these structures consist of very high-frequency terms. And as we mentioned above, the high-frequency signal will be cut in order to allow practical estimation.

Figure 3.18: Reconstructed spherical image with a different order of spherical harmonics. The first row is the original image. And the bandwidth of spherical harmonics from the second row to the last row is 1,5,10,15,20. We can see that as the bandwidth increases, the images also become clearer. And as the structure of images become more complicated, the images become hard to reconstruct.

In order to know exactly how much information the reconstructed image retained, we use again the similarity metrics MSE, SSIM, dHash, and histogram to analyze them.

From Table 3.5.2, we can see that the MSE, SSIM, and dHash all show the tendency that when the degree of the SH increases, the similarity between the reconstructed image and the original image also increased. Comparing the results between different types of images, we can find that when the image becomes complex, the similarity decreases. Images without sharp corners are also easier to reconstruct. From the similarity of histogram and Figure 3.19, we find the distribution of the pixels' value more complex to reconstruct by the SH.

Table 3.2: Similarity metrics applied on the images in Figure 3.18

Image type and degree	MSE	SSIM	dHash	Histogram similarity
inria logo 1	1940.58	0.46	0.688	0.0211
inria logo 5	1594.96	0.47	0.797	0.0214
inria logo 10	1049.38	0.57	0.844	0.1601
inria logo 15	737.20	0.69	0.875	0.2705
inria logo 20	557.40	0.76	0.859	0.3089
circle 1	2132.23	0.57	0.703	0.0526
circle 5	486.81	0.72	0.625	0.6318
circle 10	255.51	0.84	0.734	0.3916
circle 15	174.85	0.87	0.703	0.4862
circle 20	128.42	0.91	0.734	0.3435
rectangle 1	5996.08	0.34	0.593	0.9267
rectangle 5	2979.11	0.51	0.734	0.9923
rectangle 10	575.66	0.65	0.703	0.9775
rectangle 15	370.20	0.73	0.734	0.9637
rectangle 20	283.47	0.79	0.718	0.9458

3.5.3 Properties of Spherical Harmonics Functions

SH functions have many interesting properties that simplify the processing of omnidirectional images in the context of this work.

- **Orthonormal**: SH functions are not only orthogonal, but they are also orthonormal. This means the integration of the product of any different SH will return 1.
- **Rotationally Invariant**: This means that directly rotating the SH function itself is the same as feeding the SH function after rotating the input vector.
- **The integral of the product of the SH function is equal to the dot product of its spherical harmonic coefficient vector.** This means that we can consider the convolution of image and the kernel as $\int_S \hat{L}(s)\hat{t}(s)ds = \sum_{i=0}^{n^2} L_i t_i$. The integral of the product of the function collapses into a dot product of two $(n+1)^2$ dimensional vectors.

These properties provide an essential framework to build the spherical convolution, which should be shift-invariant. This property ensures the kernel will change the shape accordingly when in different positions of the sphere surface.

3.5.4 Rotation of Spherical Harmonics

Classical convolutions consist of shifting (translation) the kernel along with the image. And the spherical convolution performs shifting by rotating the spherical images. To rotate the spherical images, which uses the Euler angle decomposition. Let $g \in SO(3)$ expressed in terms of rotations R_z and R_y about the Z and Y axis respectively, g can be written as:

$$g = R_z(\alpha)R_y(\beta)R_z(\gamma).$$

The rotation follow Euler angles. Each rotation is done along the new axis generated by the last rotation. From the properties of spherical harmonics, in order to rotate the spherical signal,

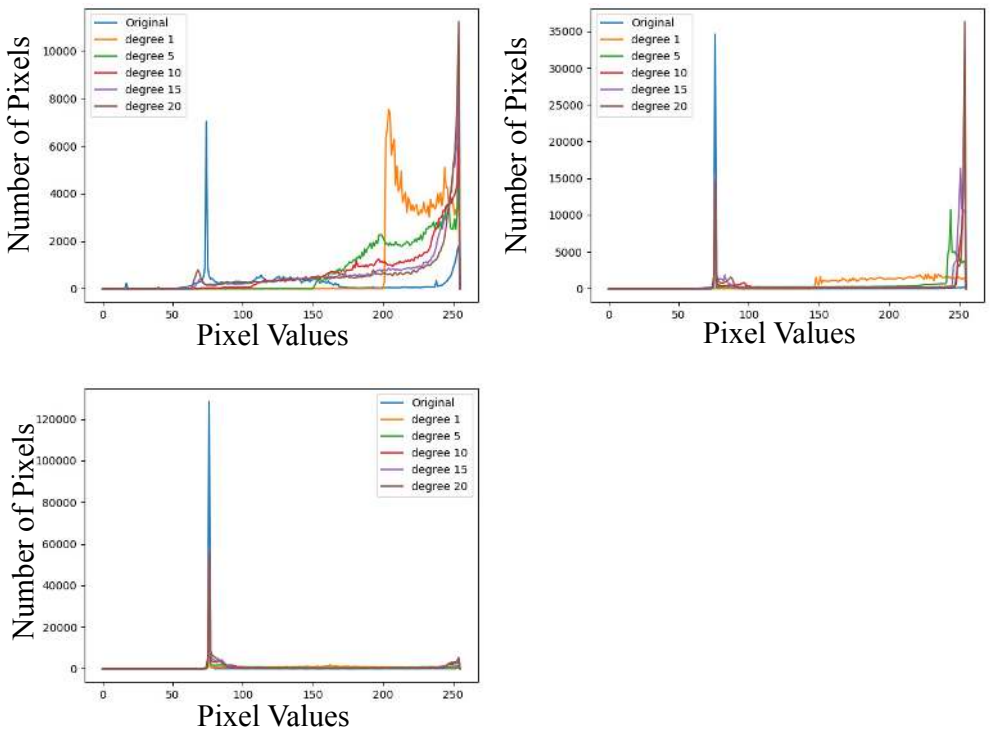


Figure 3.19: The histogram of the original image and the reconstructed ones. From top to bottom: Inria logo, circle, and rectangle image. The vertical axis presents the number of the corresponding pixel's value in an image. While the horizontal axis presents the pixels' value from 0 to 255.

we actually just need to rotate the coefficients of it. In order to apply g to the coefficient $\hat{I}_{lm}$, we note the $\Lambda(g)$ as the linear operator with the following equation:

$$\begin{aligned}\Lambda(g)\hat{I}_{lm}(S) &= \hat{I}_{lm}(g^{-1}S) \\ &= \sum_{|k|\leq l} e^{ik\alpha} d^l_{km}(\beta) e^{-im\gamma} \hat{I}_{lk}(S),\end{aligned} \tag{3.31}$$

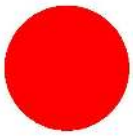
where $d^l_{km}(\beta)$ is the Wigner d-function, which explicit expression can be found in [49].

We test the rotation matrices on the equirectangular image where the center contains a red circle, and we get the results shown in Figure 3.20. We can see that as the center of the circle is located at a different place, its shape is changed. The projection from the unit sphere causes this distortion.

When the degree of the SH increases, the computation speed of the rotation matrices also become slower. To solve this problem, [23],[24] proposed recurrence relations for the rotation matrices of Real SH. [7] provided the recursive algorithm directly for complex SH. The same circle image rotated by this recursive method is shown in Figure 3.21. We can see this method can generate the same result of the Wigner D-function, with smaller computational effort than the traditional method.

3.5.5 Spectral Convolution

The spherical harmonics model the linear shift-invariant of the convolution. Another important problem still needs to be addressed, how to generate a multiscale of the filters. In flat image a filter (e.g. the Gaussian) can be simply dilated to change scale: $g(x) \mapsto a^{-2}g(a^{-1}x)$. But in the spherical one, dilating a function around the north-pole leads to overlapping at the south-pole.



(a) Original circle image

(b) Reconstruct with SH, degree is 10

(c) Shift with $\alpha = 3\pi/4, \beta = 0, \gamma = 0$

(d) Shift with $\alpha = 0, \beta = \pi/4, \gamma = 0$

(e) Shift with $\alpha = \pi/4, \beta = \pi/4, \gamma = 0$

(f) Shift with $\alpha = 0, \beta = 3\pi/2, \gamma = 0$

Figure 3.20: The equirectangular image with a red circle that rotated by different angles. From the (b) and (c), we can find that when the circle just shift along the same latitude, the shape of the circle will not change. From (d) to (f), we can find that the circle's shape will change as the latitude changed. This phenomenon is in line with our intuitive understanding.

One more problem for the implementation is about the pixel grid of the planar image has translation symmetry, which means that the filter will coincide with some areas of the image after translation (the shape and size are unchanged). But for spherical images, there is no symmetric grid like this, so before rotating the filter, the interpolation operation needs to be performed to make it have "rotational symmetry", that is, the filter will coincide with part of the image after rotation. What we should notice is the input signal is in the S^2 domain but after spherical convolution the output will be on the $SO(3)$ domain [50]:

$$(\boldsymbol{I} \star \boldsymbol{H})(g) = \int_{S^2} \boldsymbol{I}(S)\boldsymbol{H}(g^{-1}S)dS. \tag{3.32}$$

Because of the properties of the spherical harmonics, we can finally generate the discrete version of the spherical convolution:

$$(\boldsymbol{I} \star \boldsymbol{H})(g) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \sum_{k=-l}^{l} \hat{\boldsymbol{I}}(l,m)\hat{\boldsymbol{H}}(l,k)D^l_{k,m}(g). \tag{3.33}$$

In the next section we will use the Gaussian filter as an example to show how the spherical convolution on the spectral domain works.

(a) Shift with $\alpha = 3\pi/4, \beta = 0, \gamma = 0$

(b) Shift with $\alpha = 0, \beta = \pi/4, \gamma = 0$

(c) Shift with $\alpha = \pi/4, \beta = \pi/4, \gamma = 0$

(d) Shift with $\alpha = 0, \beta = 3\pi/2, \gamma = 0$

Figure 3.21: The circle image rotated by the rotation matrices generated by the recursive method. We can find that the results are exactly the same as the corresponding rotation angle in Figure 3.20.

3.5.6 Results of Convolution with Spherical Gaussian

By solving the spherical PDE as mentioned above, we get the coefficient of spherical Gaussian:

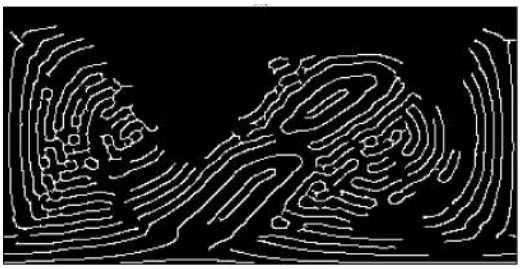
$$\widehat{G(t)}_{lm} = \begin{cases} \sqrt{\dfrac{2l+1}{4\pi}} e^{-l(l+1)t} & if\ m = 0 \\ 0 & else \end{cases} \tag{3.34}$$

To test the spherical Gaussian filters smoothing effect, we applied it to an image with noise. The results that applied the spherical Gaussian filter are shown in Figure 3.22. In the subfigure (b), we can see that the noise and the reconstructing lead to the ripple noise. After spherical Gaussian smoothing, the image still keeps the main structure of the object, and it is also denoised. So the spherical Gaussian shows the desired processing effects. In order to prove the spherical convolution is necessary, we generate a synthetic texture of the circle in different positions of the sphere in the equirectangular plane. And we process it with the planar Gaussian and spherical Gaussian filters. These results are shown in Figure 3.23. We can notice that the planar Gaussian filter cannot keep the projection's deformation, which means when we project back to the sphere surface, results in artifacts. The spherical smoothing keeps the correct shape of the objects. So the spherical convolution is necessary for the omnidirectional image processing.

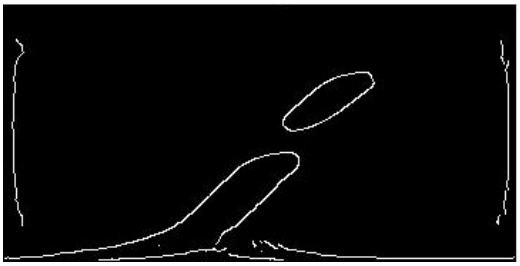
(a) Original image with white noise

(b) Reconstruct with SH, bandwidth=30

(c) Spherical Gaussian smooth ($t = 0.05$)



(d) Canny edge detect of the reconstruct image



(e) Canny edge detect of the smooth image

Figure 3.22: The spherical Gaussian smoothing results. (a),(b) and (c) present smoothing procedures. To show the smooth results more clearly, we also show the edge detect the result of the image in (d) and (e).

(a) Synthetic texture of the circles (equirectangular plane)

(b) Smoothing with the flat Gaussian filter

(c) Smoothing with the spherical Gaussian filter

Figure 3.23: Examples of flat Gaussian convolution and spherical Gaussian convolution. (b) uses a 3×3 kernel while (c) use the equivalence size spherical kernel. From (a), we can find that when the circles get close to the poles, the circles will be deformed. In (b), the flat Gaussian filter can not keep this deformation, while the spherical one kept.

3.6 Conclusions

In this chapter, we presented the background on omnidirectional images. We discussed how to represent and how to define the adapted processing operators. We tested different camera model calibrations and image processing in the unit sphere. We also analyze the results of these image processing methods to have a deeper understanding of their characteristics. These models and methods are the basis for correctly processing and using the information inside omnidirectional images, which provide the essential functions to construct our inpainting approach in the following chapters.

Chapter 4

Inpainting of Panoramas from Reconstruction Networks

In order to design an effective generative model for omnidirectional inpainting, we first build models and evaluate their performance with a bottom-up complexity scheme: reconstruction, inpainting from denoising, and then inpainting from deep generative networks. Following this bottom-up model design scheme, we can evaluate the performance of our model step by step. In this chapter, we focus on the reconstruction and inpainting from denoising. Reconstruction is the procedure of extracting the features from data and then recovering the data from these features. It can assess the model's ability to extract information. Then inpainting from denoising requires to recover the missing information of the input data. It can further evaluate the inference ability of the model. Based on these, we can finally construct our generative model for inpainting.

As we mentioned in the previous chapter, there are two adaptation strategies of convolutional operators for omnidirectional images: one uses the geodesic metric in the sphere to redefine the convolutional kernels [13], while the other one uses the Fourier transform to process the signal in frequency domain [3]. Since these two methods process the images on equirectangular and spectral domain respectively, we name them equirectangular convolution and spectral convolution. To obtain high-quality image reconstruction results of the spectral convolution requires a high bandwidth, which means cost many computing resources. Comparing to the expensive computing cost of the spectral convolution, the equirectangular convolution can store the positions of the spherical neighbors (projection map) at the first computation and then reuse it in the following convolutions. Considering limited computing resources, we selected equirectangular convolution in the design of our adapted method.

4.1 Inpainting Problem Formulation

Inpainting is the problem of generating improved images but with similar content of a set of given images with missing or damaged regions. We can use different similarity metrics to formulate the inpainting. The typical choice of the similarity metric is the l_2 distance:

$$\mathcal{L}_2 = \left\| x - I(x) \right\|, \tag{4.1}$$

where x is the original image, $I(x)$ is the inpainting result. We can also employ the l_1 metric for obtaining sharper reconstructions. However, these metrics only focus on the similarity on the pixel's level, which leads to blurred results of the inpainting parts. To tackle this problem, perceptual metrics were proposed. Perceptual metrics compute the similarity from the features of images instead of directly using pixels. We can define the general formula of the perceptual metrics as:

$$\mathcal{L}_{feat} = \left\| C(x) - C(I(x)) \right\|_2, \tag{4.2}$$

where C is a function, which extracts the features of the images. For instance, we can employ classical feature extraction methods such as scale-invariant feature transform (SIFT), the histogram of oriented gradient (HOG), and local binary patterns (LBP). Also, recently neural networks are used to extract the features of images. Learning-based perceptual similarity functions become more and more used, since they can provide better-extracted features than the classical ones, such as the Learned Perceptual Image Patch Similarity (LPIPS) [59].

For the application of removing objects, since the inpainting results would not be the same as the input, the objective function focus on the similarity of the distribution of the images:

$$\mathcal{L}_{dis} = \left\| D(x) - D(I(x)) \right\|_2 , \tag{4.3}$$

where x is the input images, and D is the function to compute the distribution of the images. Because the distribution of images is hard to describe, the function D is generally learning-based . The are many popular distribution metrics such as the Fréchet inception distance (FID), maximum mean discrepancy (MMD), and the Wasserstein critic.

4.2 Reconstruction for Omnidirectional Images

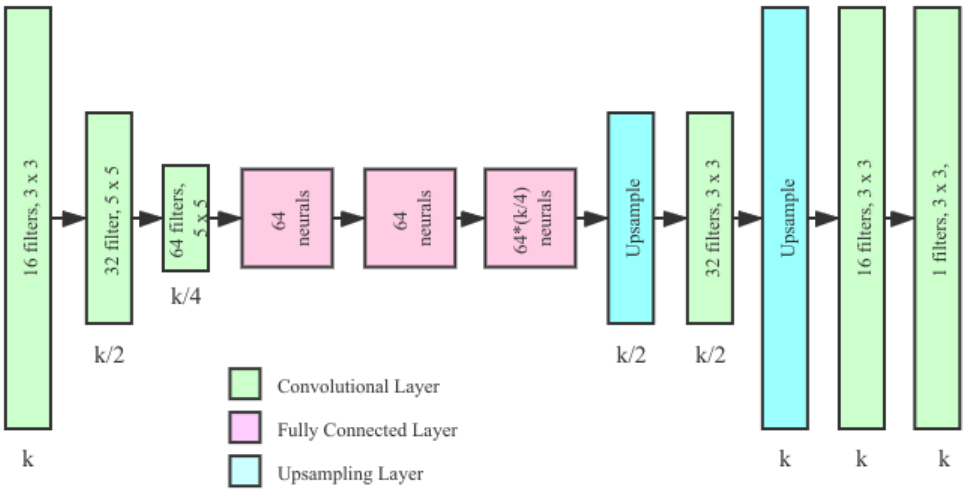
In this section, we show how to design a convolutional autoencoder to perform the reconstruction task on omnidirectional images. Autoencoders can be seen as an unsupervised approach for learning feature representation from unlabeled training data. It is constituted by two main parts: an encoder that extracts the feature (code) from the input, and the decoder that exploits the code to reconstruct the original input as closely as possible. The compressed code is the high dimensional representation of the input data, also called the latent variable. The space that contained all the compressed code is latent space. Autoencoder can only learn the relationship between the code and the images, but cannot generate the new code. So autoencoder is not a generative model, because it cannot generate the new samples.

4.2.1 Reconstruction of Spherical Digits

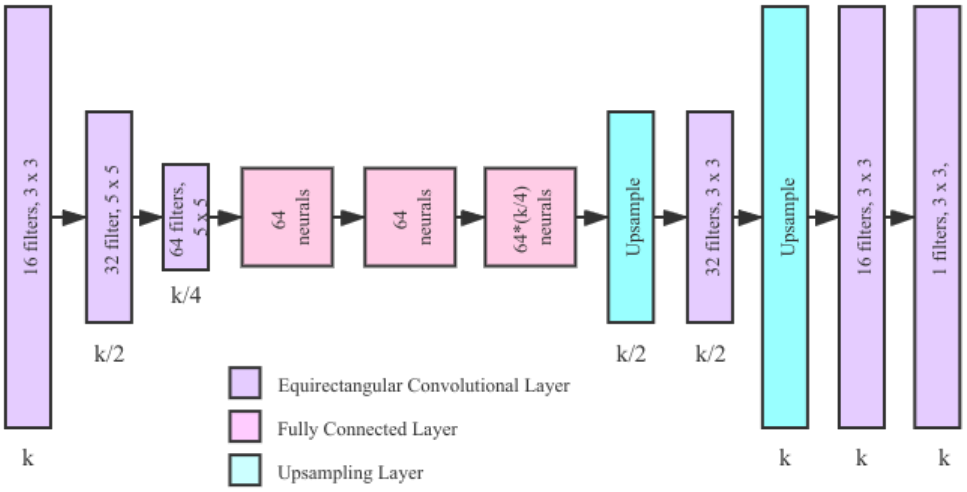
Based on the definition of autoencoders, we build a standard and efficient autoencoder. The encoder consists of three convolutional layers and one fully-connected layer. The convolutional layers extract the features while the fully connected layer combines and compresses the code. The other left layers make up the symmetric structure decoder. The structures of these autoencoders are shown in Figure 4.1. In the adapted autoencoder, we replace the traditional convolutional layer with equirectangular ones. In order to train these networks for performing inpainting from reconstruction, we used the mean squared error (MSE) as a loss function. Since the autoencoder's objective is to increase the similarity of input and reconstructed output.

Spherical MNIST Dataset

To get fast feedback, we decide to reconstruct the digits from the MNIST dataset, which contain much simpler information than the real images. The work of [8] provided a spherical MNIST dataset. They projected the digits images onto the sphere using stereographic projection and then map them back to the (θ, ϕ) plane. Furthermore, these spherical images are rotated randomly to simulate a more general situation. Examples of this dataset are shown in Figure 4.2. Compared with some real spherical image datasets, this dataset has a simpler image structure and many more samples. So it can help us to have a fast and sufficient test for the framework.



(a) Classical Convolutional Autoencoder



(b) Adapted Convolutional Autoencoder

Figure 4.1: Structure of the convolutional autoencoder and the adapted one.

Results of Reconstruction

We trained the original autoencoder and the adapted one with the whole spherical MNIST training set, which contains 60000 images. We trained them for 100 epochs with the Adam optimizer. The variation of the training loss is shown in Figure 4.3. We can find that in the first ten epochs' training, the loss drops sharply. And after that, although the loss decreases, the trend becomes steadily. When we check the testing results, we also find the same appearance. After ten epochs training, we can get good enough results, and after that epoch, we cannot find noticeable improvement. We randomly choose ten digits from the testing set to perform the reconstruction with the models saved at ten epochs. The visual results are shown in Figure 4.4.

From these results, we can observe that both the original autoencoder and the adapted one can reconstruct the spherical digits well after ten epochs' training. Although both autoencoders' performance seems very close to human visual perception, there are some differences in details of the reconstructed images from the two autoencoders. For example, the third column images

Table 4.1: Evaluation of reconstruction (digit numbers) with the SSIM and LPIPS metrics.

	SSIM[1]		LPIPS [2]	
Epochs	Original	Adapted	Original	Adapted
5	0.534	**0.548**	0.1327	**0.1285**
10	0.735	**0.758**	0.0823	**0.0767**
15	0.754	**0.762**	0.0789	**0.0758**
20	0.748	**0.764**	0.0791	**0.0754**
	[1] *Better closer to 1.*		[2] *Better closer to 0.*	

are the horizontal 3, and the original autoencoder wrong link the head of 3 while the adapted one reconstruct it correctly. The other column images also show some similar errors in the original autoencoder output, but not as clear as the third one. In sum, from the human eye, we still can find improvements in the adapted autoencoder compared to the original one, but improvements are not noticeable.

Quantitative Evaluation

We employed the structural similarity index measure (SSIM) and Learned Perceptual Image Patch Similarity (LPIPS) metric [59] to measure the performance of each reconstruction quantitatively. SSIM and LPIPS are the metrics to evaluate the similarity of the images. SSIM focuses on the structural similarity while the LPIPS focuses on the perceptual similarity, and it is a learning-based metric. For SSIM, a higher value means more similar, while the LPIPS is the opposite. We use the whole testing set to generate its corresponding reconstructed images and then compute the SSIM and LPIPS. We computed the average values of SSIM and LPIPS for every five epochs. As we mentioned above, after around ten epochs, the results for both autoencoders did not improve too much. So we only evaluated the first 20 epochs' results. The evaluation results are presented in Table 4.1. In all epochs, both the SSIM and LPIPS show that the adapted autoencoder performs better than the original one when reconstructing the spherical digits. However, the improvements are limited, as the gap of the evaluation values in the same epochs are small. The results of the evaluation metrics go accordingly to the conclusions we got.

4.2.2 Reconstruction of Real Panoramas

Although reconstructing spherical digits proved the equirectangular operators could improve the performance when representing panoramas, the gap is small. One possible reason is that it is too easy for the autoencoder to reconstruct the digits. We then tried to directly use the above autoencoder to perform the reconstruction on the real spherical images. But the obtained

Figure 4.2: Comparing between flat digits and spherical digits. The first row digits come from the MNIST dataset while the second-row images are the corresponding rotated spherical digits.

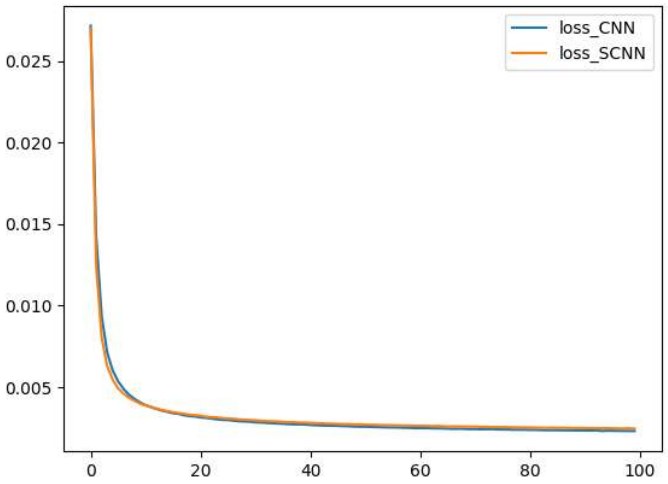


Figure 4.3: Training losses for reconstructing spherical digits. The loss_CNN presents the variation of the convolutional autoencoder's training loss, while the loss_SCNN the variation of the adapted autoencoder's training loss. From the tendency of the losses, we can roughly estimate that the two autoencoders have the similar performances.

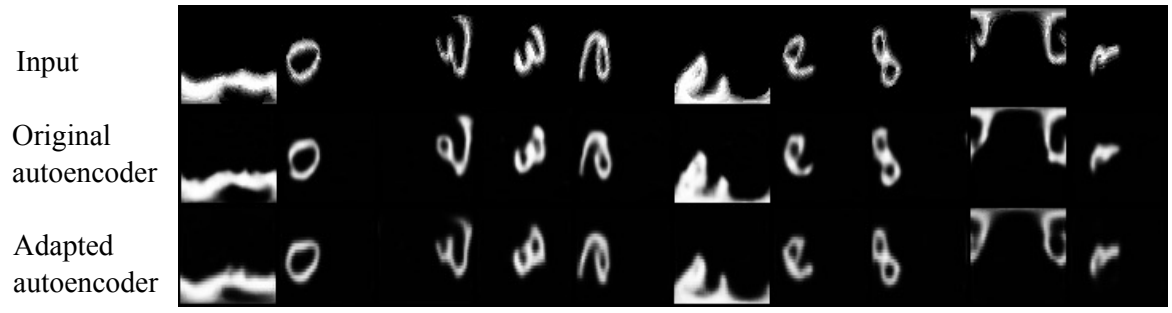
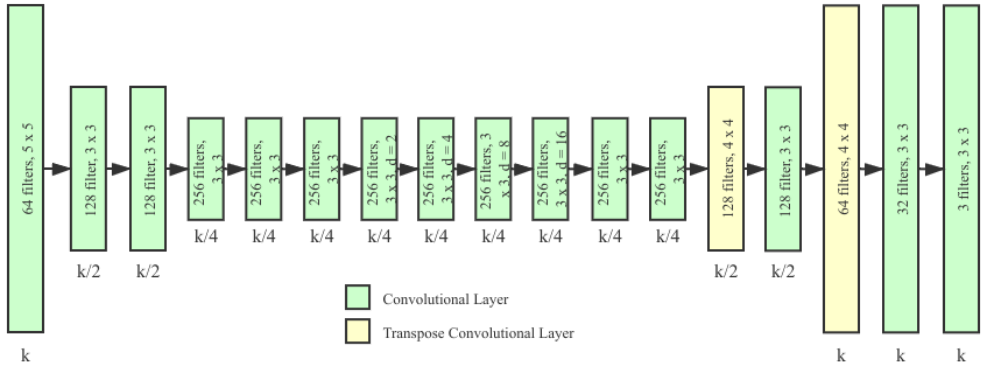
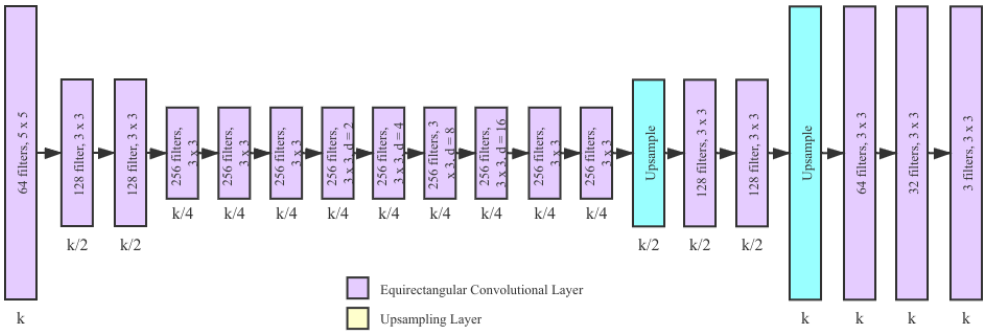


Figure 4.4: Reconstruction of the spherical digits. The first row images are the inputs. The second-row images are the reconstructed results of the original autoencoder while the bottom ones come from the adapted autoencoder.



(a) Original Convolutional Autoencoder



(b) Adapted Convolutional Autoencoder

Figure 4.5: The structure of the autoencoders. The d in several blocks means the dilation of convolution. For the adapted autoencoder, we replace all the convolution layers to equirectangular ones. In the convolution transpose layers to the combination of upsampling and equirectangular convolutions.

results showed that both the classical autoencoder and the adapted one could not handle the real panoramas well. We designed a more elaborated model, which selected the Globally and Locally Consistent Image Completion (GLCIC) network as the backbone to perform the reconstruction.

Architecture of Autoencoder

This model only has the convolutional layers and convolutional transpose layers except for the activation layers and the batch normalization layers, as shown in Figure 4.5(a). To present the network concisely, convolutional layers and convolutional transpose layers are a combination of convolution, batch normalization, and activation.

Similar to the previous model, the adaption of this model to omnidirectional images, replacing all the convolutional layers in the autoencoder with the equirectangular one. The original network uses convolution transpose layers in the decoder part. However, this layer makes the uneven overlap in the deconvolution, mainly when the kernel size is not divisible by the stride. The uneven overlap is a phenomenon that puts more of the metaphorical paint in some places than others. The method to solve this problem is to decompose convolution transpose layers as upsampling followed by a convolution to compute the feature in higher resolutions. Therefore, we use the upsampling layer to resize the data (using either one or the other bilinear to interpolation) and then add a convolutional layer, replaces the convolution transpose layers. The final adapted network is shown in Figure 4.5(b). The loss function we still use the MSE loss.

(a) Original Panoramas

(b) Panoramas rotated along y axis with 315 degrees and rotated along z axis with 17.5 degrees.

(c) Panoramas changed the brightness, corresponding to the panoramas of (b)

Figure 4.6: Data augmentation of the spherical image. We applied the rotation and brightness change on the same image to get these results.

Real Spherical Images Dataset

To perform high-quality inpainting, the network needs a lot of data and epochs to train, making it hard to get fast feedback. So we decide to do the data augmentation of just one image to decrease the difficulty of reconstruction. The augmentation methods are rotation and brightness change. The rotation here means to change the viewpoint of the unit sphere. So after the rotation, the images still follow the geometry relation of the unit sphere. To keep the balance of the image types, we organized the images with constant brightness and kept the same numbers for each brightness. Finally, we generate 200 images as the training set and 40 images as the test set. Some examples are shown in Figure 4.6.

Reconstruction Results

To compare fairly, all the parameters were kept the same for both autoencoders. We train the networks for a total of 250 epochs. The variation of the training loss is shown in Figure 4.7. As it can be observed, it has a huge drop after the first epoch's training, and then the loss slowly decreases. The reconstruction results for every 50 epochs are shown in the Figures 4.8, 4.9. The reconstructed images are from the test set. We can find that although the training loss becomes small, the reconstructed borders of the objects at 250 epochs are still not well reconstructed. This is because the MSE loss is computing at the pixel level, which makes it unable to handle the structure well for the images that contain complex information.

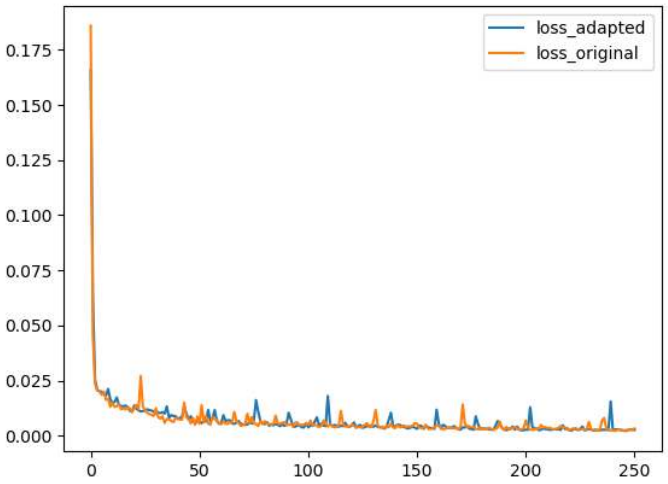
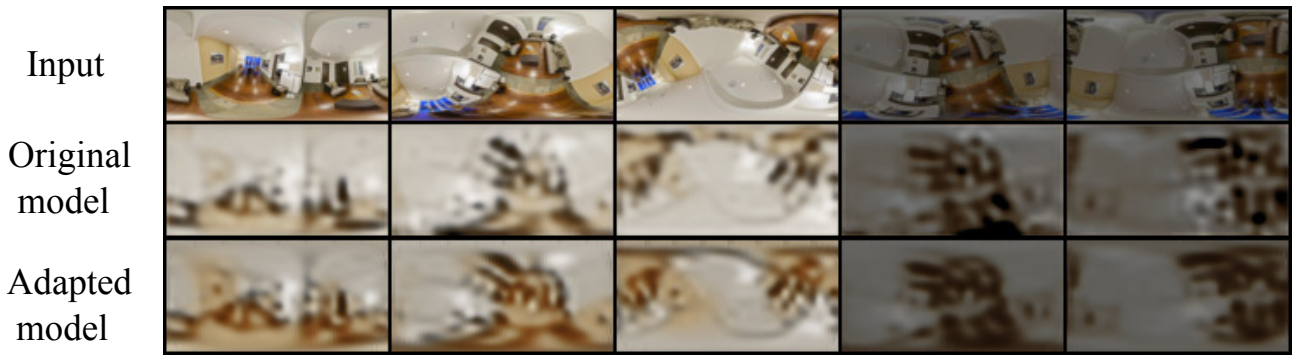
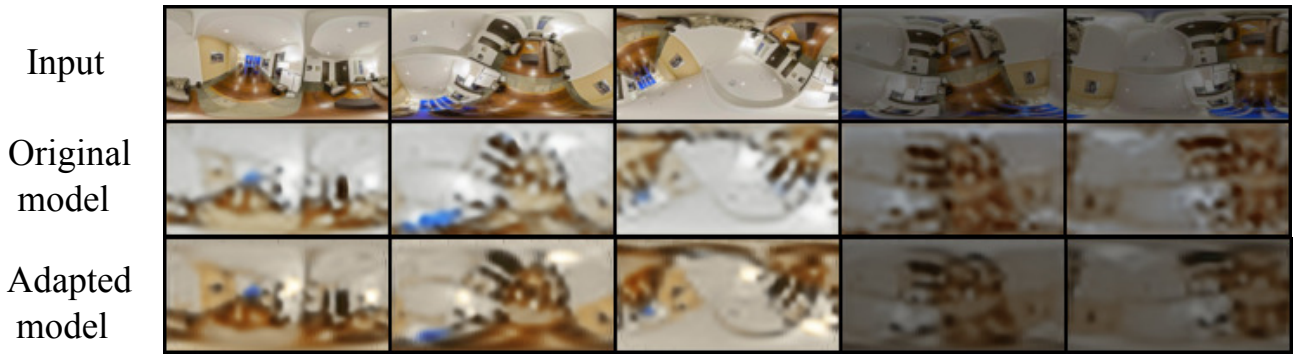


Figure 4.7: Training losses of the original autoencoder and the adapted one. From the tendency of the losses, we can roughly estimate the performance of the two autoencoders.

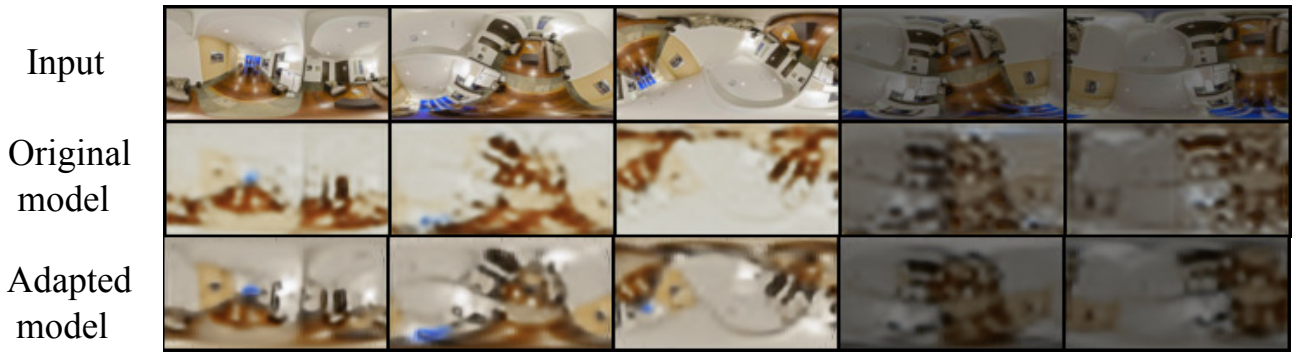


(a) Reconstructed results of the original model and the adapted one at 50 epochs

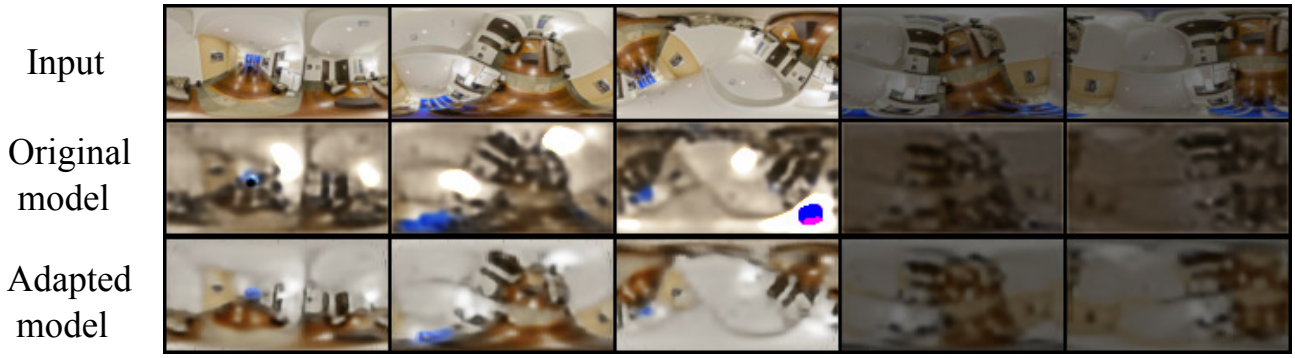


(b) Reconstructed results of the original model and the adapted one at 100 epochs

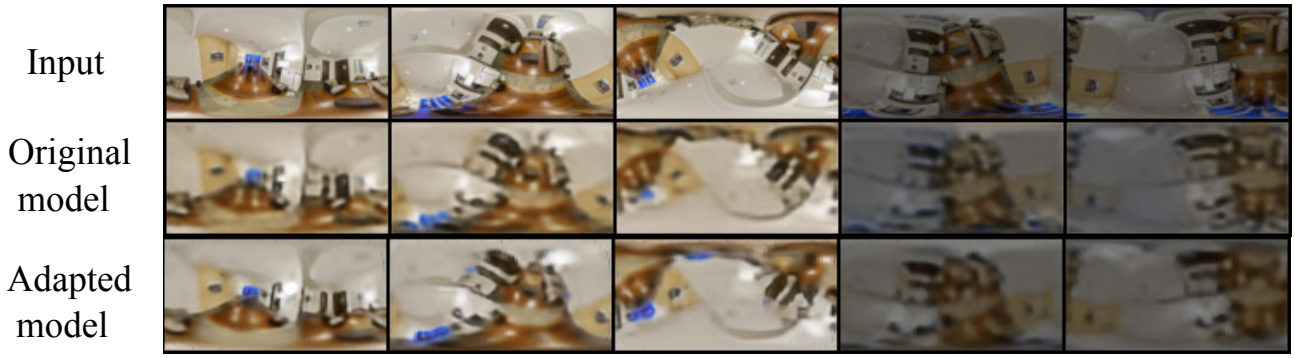
Figure 4.8: Part I: Reconstruction results of both the original generator and then adapted one to equirectangular images. The results come from the test set images after training in different epochs. For each group, the first row is the input image. The second row is the output of the original autoencoder while the third row is the output of the adapted one.



(a) Reconstructed results of the original model and the adapted one at 150 epochs



(b) Reconstructed results of the original model and the adapted one at 200 epochs



(c) Reconstructed results of the original model and the adapted one at 250 epochs

Figure 4.9: Part II Reconstruction results of both the original model and the adapted one to equirectangular images. The results come from the test set images after training in different epochs. For each group, the first row is the input image. The second row is the output of the original autoencoder while the third row is the output of the adapted one.

From the results, we can see that both the original and adapted autoencoder improve the performance as the training epochs increased. For instance, after 50 epochs' training, the blue windows completely disappeared from both outputs. And then at 100 epochs, they appeared, but their structure is still blurry. Finally, at 250 epochs, although we can easily find the differences between the inputs and outputs, they are much closer to inputs than at 100 epochs' results. Comparing the trained model's performance for these epochs, we can find that the adapted reconstruction model has better performance to reconstruct the panoramas than the original model before 250 epochs. We can see that before the 250 epochs, the reconstructed colors of objects from adapted autoencoder are closer to the input images than the original one. But at epochs 250, the original autoencoder has a close performance as the adapted one.

Quantitative Evaluation

We evaluate the reconstruction results with the SSIM and LPIPS metrics in the same way as done for the reconstruction of digits. These metrics are shown in Table 4.2. We can see that both the results of SSIM and LPIPS are similar to our intuitive feeling. The images generated by the adapted model have a smaller difference to the input images. But except for the models at 150 and 200 epochs, it does not have too much improvement (around 10%). Comparing computational aspects, the original model takes on average 1.89s per epoch to process 200 images, while the adapted required 15.76s per epoch.

In sum, these results prove that, to a certain extent, the equirectangular model can improve the performance for reconstructing omnidirectional images. However, the improvement is limited and has the issue of extra computing cost for the equirectangular convolution.

Table 4.2: Evaluation of reconstruction (real images) with the SSIM and LPIPS metrics.

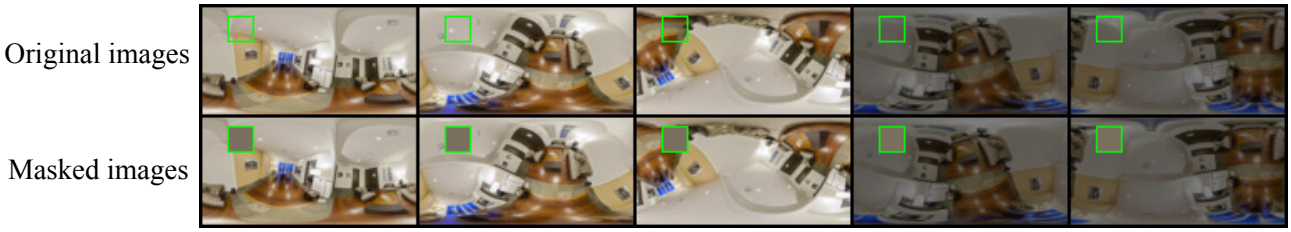
	SSIM[1]		LPIPS [2]	
Epochs	Original	Adapted	Original	Adapted
50	0.442	**0.485**	0.444	**0.412**
100	0.548	**0.562**	0.357	**0.315**
150	0.504	**0.625**	0.364	**0.291**
200	0.518	**0.614**	0.424	**0.312**
250	**0.637**	0.631	0.315	**0.297**
	[1] *Better closer to 1.*		[2] *Better closer to 0.*	

4.3 Inpainting from Denoising with Convolutional Autoencoder

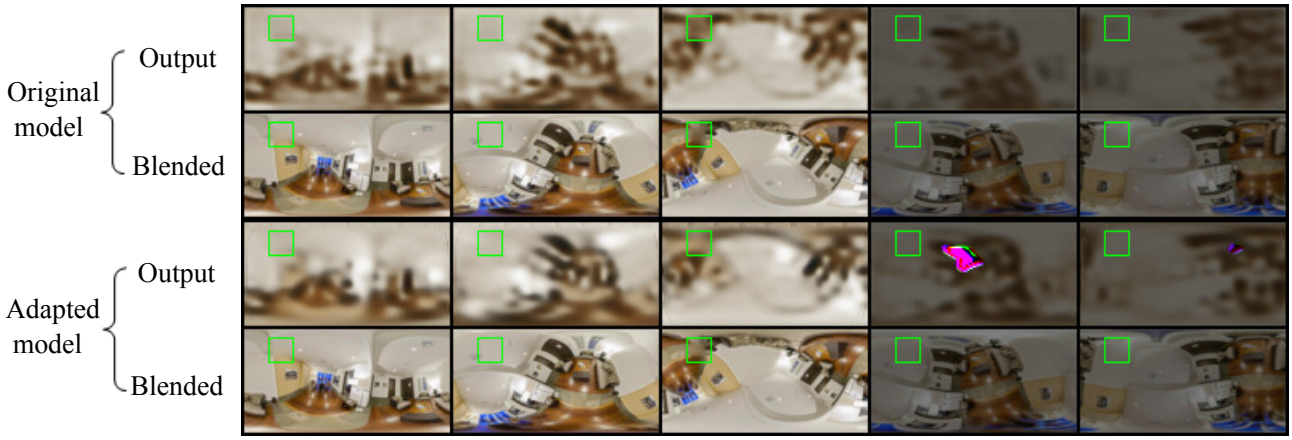
After we successfully performed the reconstruction with the adapted autoencoder, we can perform the image inpainting. Inpainting is a harder task than reconstruction. Since the input information is not complete, the network needs to infer the missing information from the existing ones. Inpainting can further challenge the information extraction and utilization capabilities of the model. We should notice that this inpainting is still not the generative inpainting yet. Because these models only know how to use the code to present the corresponding images, and they cannot draw a new code from the latent space to present new images. Thus, they can only restore the images that have a similar code as the training data. Our dataset was built using data augmentation from images of a single indoor scene. So the test and training sets have similar codes, which can still be used in this task.

4.3.1 Architecture and Loss Function

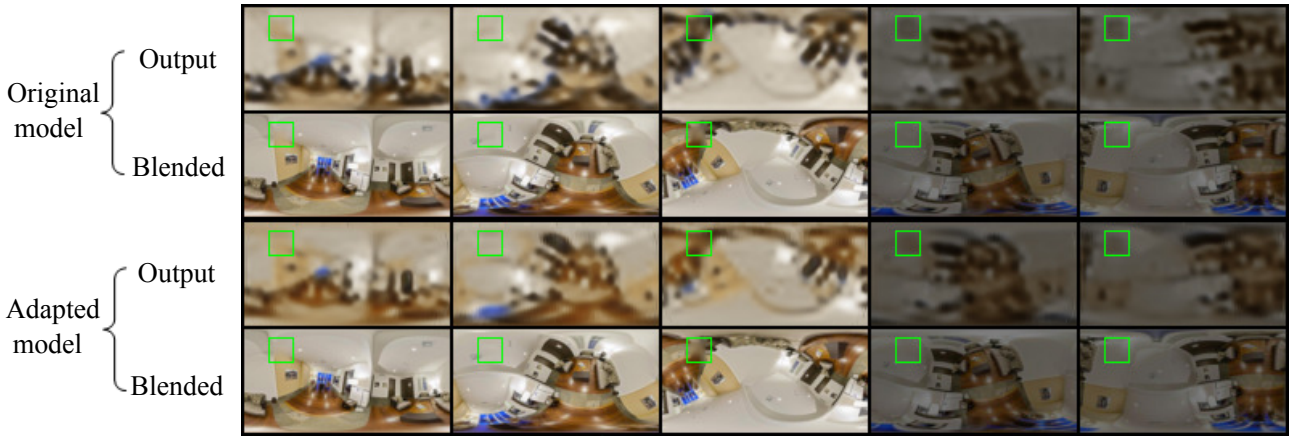
The architecture of the autoencoders is the same as the ones shown in Figures 4.5. There are two differences between reconstruction and inpainting from denoising tasks. Firstly, we generate images with missing regions, as shown in Figure 5.5. The main goal of the inpainting from denoising is to recover the original images. i.e., the model needs to fill one missing area. And then, the pixels value of the corresponding position of the masks will be replaced with the mean of the whole dataset's pixels value. The loss function of this inpainting is to make the mean square error with different weighting pixels. The missing parts have higher weights, while the non-masked parts have lower weights.



(a) Original images and the masked ones (actual input) used for training the model for inpainting



(b) Inpainting results of original autoencoder and the adapted one at 50 epochs.

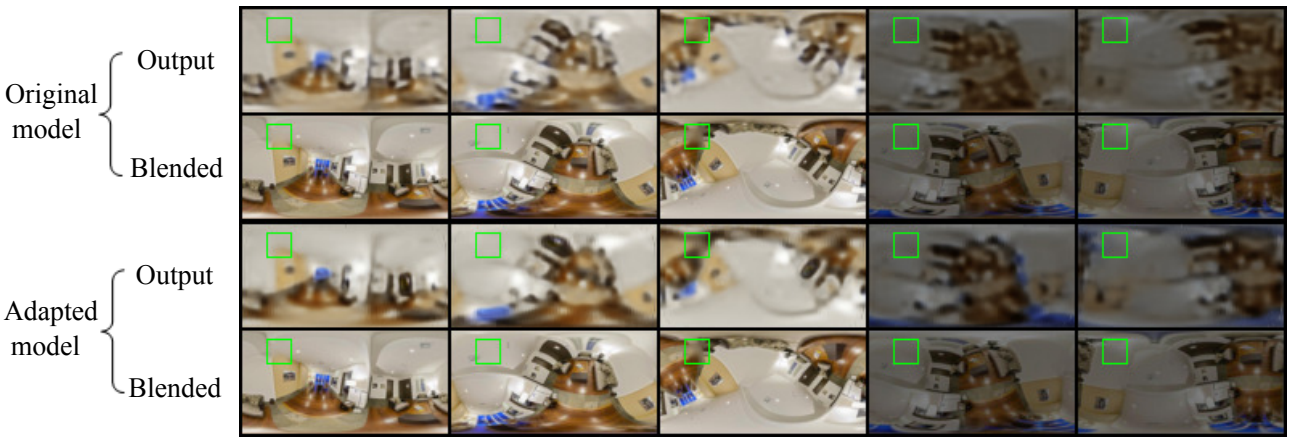


(c) Inpainting results of original autoencoder and the adapted one at 100 epochs.

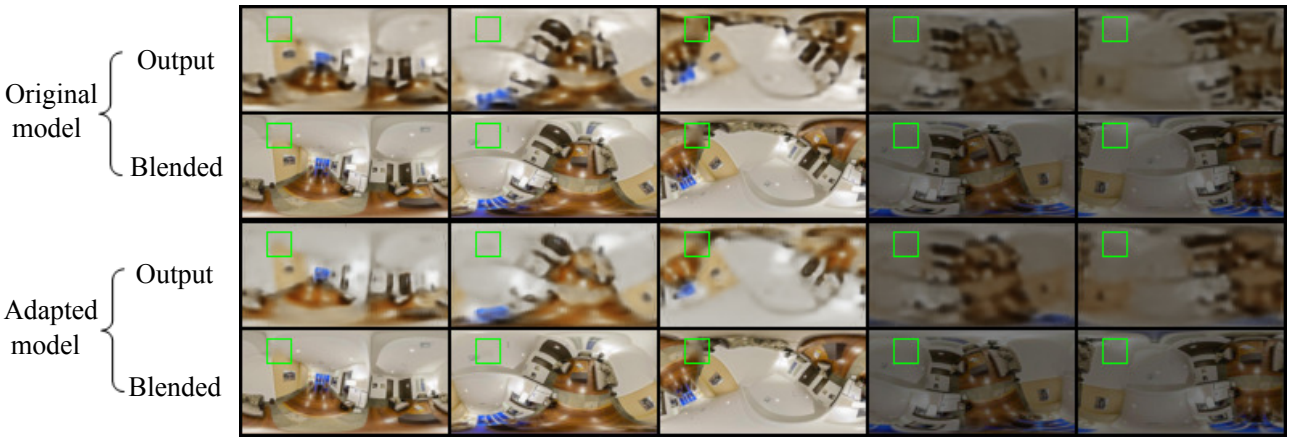
Figure 4.10: Part I: Results of inpainting for the missing regions. The input images come from the test set. The first row of (a) is the original images, while the second row is the masked images. The first and third row images of (b), (c) are the output of the original and adapted inpainting model. In contrast, the second and fourth row of images comes from the blend of the missing areas in the masked images and the corresponding areas in the output images.

4.3.2 Results of Inpainting

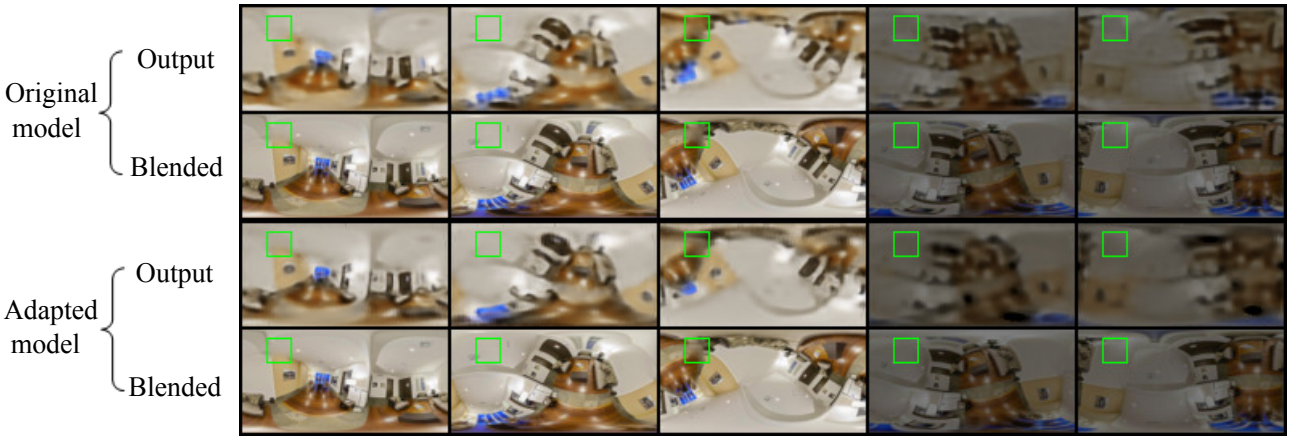
For the test stage, we chose five fixed regions to perform the inpainting, shown in Figure 5.5. And these five areas are located at different places of the image, which can comprehensively reflect the inpainting performance. Every time, we just restore one indicated region. We replace the pixels in the inpainting area with the mean pixels' value of the whole dataset. The generative model is asked to perform inpainting on these corrupted areas.



(a) Inpainting results of original generator and the adapted one at 150 epochs



(b) Inpainting results of original generator and the adapted one at 200 epochs



(c) Inpainting results of original generator and the adapted one at 250 epochs

Figure 4.11: Part II: Results of inpainting for the missing regions. The input images come from the test set. The first row of (a) is the original images, while the second row is the masked images. The first and third row images of (b), (c) are the output of the original and adapted inpainting model. In contrast, the second and fourth row of images comes from the blend of the missing areas in the masked images and the corresponding areas in the output images.

We present the one reconstruction typical block. This block's position close to the pole, which has the largest distortion. The missing areas in the masked images blended with the corresponding areas in the reconstructed images to get the final inpainting results. The inpainting results are shown in Figures 4.10, 4.11. The green rectangles indicate the inpainting areas. We can see that the output images are more blurry than the reconstruction ones. The input images

Table 4.3: Evaluation of inpainting results with MSE and LPIPS.

	MSE[1]		LPIPS [2]	
Epochs	Original	Adapted	Original	Adapted
50	0.0127	**0.0115**	0.241	**0.209**
100	0.0118	**0.0113**	**0.187**	0.190
150	0.0106	**0.0101**	0.182	**0.163**
200	**0.0094**	0.0103	**0.152**	0.175
250	**0.0087**	0.0096	**0.149**	0.173
	[1] *Better closer to 0.*		[2] *Better closer to 0.*	

lost some information in the inpainting task, which makes the reconstruction problem harder. From visual inspection, we cannot easily indicate which autoencoder provided better results, since both models produced the blurred results.

4.3.3 Quantitative Evaluation

In the inpainting evaluation, we only need to compare the obtained results in the missing parts. For that, we adapted MSE and the LPIPS to quantitative results. For every 50 epochs, we compute the MSE and LPIPS for the five inpainting areas of the whole testing set. And then, we compute the average value of the MSE and LPIPS for each model. These results are shown in Table 4.3. At 50, 100, 150 epochs, the adapted model performs better in the MSE metric than the original one, while at 200 and 250 epochs, the original one performs better. For the LPIPS metric, except for the 50 and 150 epochs, the original one has better performance than the adapted one. But we should notice that the gaps between the two models are quite small. From the results, we can see that in the inpainting, the adapted model presented still a close performance to the original one.

Chapter 5

Inpainting with Generative Models

In this chapter, we present an inpainting strategy with generative neural networks, exploiting the concepts and formulations discussed in the previous chapters. In the previous chapter, we have shown that convolutional network models are capable of processing and reconstructing images with missing information. These models are the basis of constructing the generative inpainting model for inpainting omnidirectional images.

5.1 Introduction to Generative Models

The code extracted from a image by autoencoder is the lower dimensional presentation of the images, which also called the latent variable because we cannot observe it directly. We should notice that although the latent variables are discrete points, they obey a continuous distribution, which means we can generate new latent variables not exist in the training dataset but obey the same distribution. And all these latent variables form a continuous latent space.

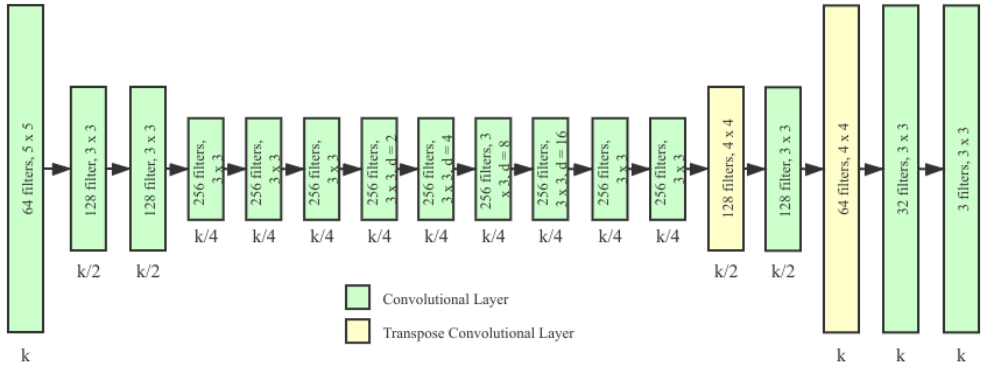
The models we presented in the previous chapter only know how to extract and utilize the latent variables of the images that are in the training dataset. But they cannot learn the distribution from the latent variables and draw new code also obey this distribution. This limitation hugely decreases the scope of application of these models. This is a major reason why generative models are often preferred in image synthesis tasks, including inpainting. Generative models can learn the distribution of the given training dataset, and generate new samples which obey the same distribution. The two most adopted learning strategies to obtain generative models are variational autoencoders (VAEs) and generative adversarial networks (GANs).

Sometimes the distribution of the dataset is very complex, and we cannot give an explicit definition of the model distribution to approximate it. GANs use the implicit density estimation, which takes the game-theoretic approach: learn to generate from training distribution through a two-player game. GANs frame the problem as a self-supervised learning problem with two sub-models: a generator that we train to synthesize new examples, and a discriminator that tries to classify examples as either real (from the domain) or fake (generated). The objective function of the GAN is defined as:

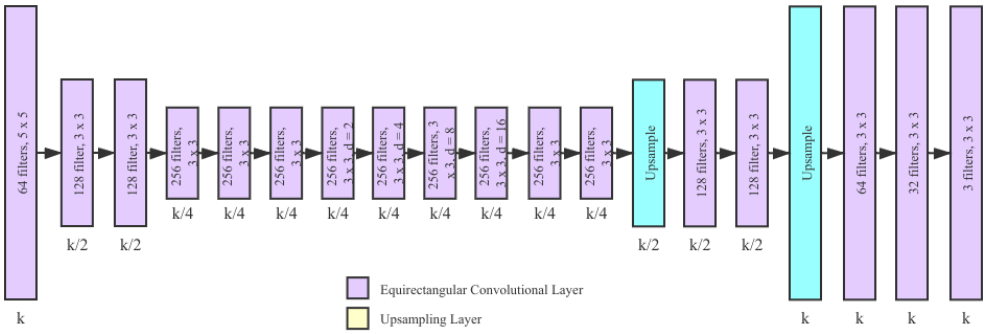
$$\min_D \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))], \tag{5.1}$$

where D denotes discriminator and G denote generator. In the above equation, if the input to the discriminator comes from true data distribution, then D(x) should output close to 1 while from $G(z)$, $D(G(z))$ should output close to 0 to maximize the objective function. While the generator wants to make $D(G(z))$ output close to 1 to minimize the objective function. From this objective function, we can design more elaborated losses to train the generator and discriminator.

We used models that perform the inpainting from denoising as the generators since they have been proved to have enough ability to process the latent variables. And then, we combine



(a) Original Convolutional Autoencoder



(b) Adapted Convolutional Autoencoder

Figure 5.1: The structure of the autoencoders. The d in several blocks means the dilation of convolution. For the adapted autoencoder, we replace all the convolution layers to equirectangular ones. In the convolution transpose layers to the combination of upsampling and equirectangular convolutions.

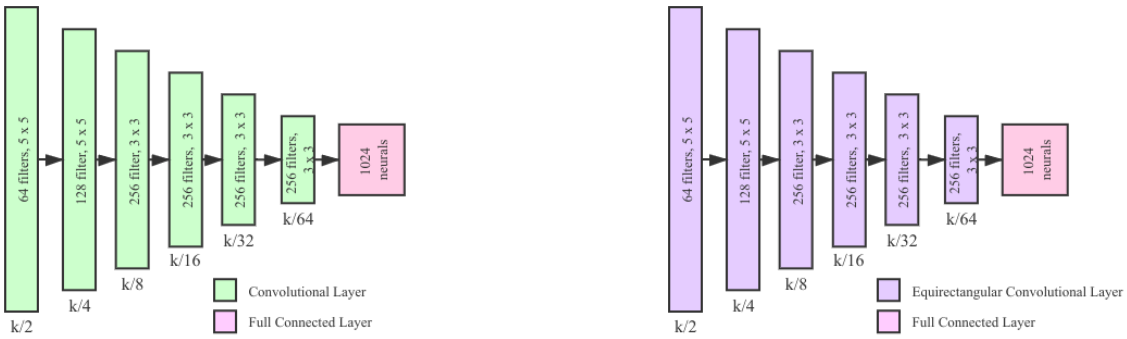
these generators with the discriminators to teach them to learn the distribution to produce restored panoramic images. How this model performs the inpainting is described in the following sections.

5.2 Generative Inpainting Model

The structure of the original generator and the adapted one are shown in the Figure 5.1, which are the same structure as the autoencoders we showed in the previous chapter. After the completion of the images, the global discriminator takes the whole image as input, while the local one only takes the local areas with missing information. Finally, both discriminators generate a vector and combine to give the joint judgment about whether the image is generated. The original structure of the discriminator is shown in Figure 5.2(a). We build our discriminator by combining the equirectangular operators. The structure of the modified networks for panoramas is shown in Figure 5.2(b).

5.2.1 Losses Functions

The model needs to understand the content of an image and produce a plausible hypothesis for the missing parts. However, this task is inherently multi-modal as there are multiple ways to fill a missing region while also maintaining coherence with the global context for the images.



(a) Discriminator structure

(b) Adapted discriminator structure

Figure 5.2: The structure of discriminators. The global and local discriminators have the same structure in (b). The difference is only the input shape.

Generally, we decouple this burden in the loss function by jointly training a generator to minimize both a reconstruction loss and an adversarial loss. The reconstruction loss captures the overall structure of the missing region concerning the context, while the adversarial loss has the effect of picking a particular model from the distribution. We selected the mean squared error (MSE) as the reconstruction loss to ensure the difference on the pixels level, which makes the reconstructed part tend to get the blurry result. Solving this problem adds the adversarial loss, which uses the binary cross-entropy loss (BCE loss) function:

$$\begin{aligned} L(x,y) = l_1, \ldots, l_N, l_n = -(y_n \log x_n + (1-y_n)\log(1-x_n)), \\ \mathcal{L}_{BCE}(x,y) = \left\| L(x,y) \right\|, \end{aligned} \tag{5.2}$$

where y_n is the ground truth label while the x_n is the predicted label from the discriminator. We denote $D(x, M)$ as the output of discriminator. And the final adversarial loss to train the discriminators is:

$$\mathcal{L}_{adv} = \mathcal{L}_{BCE}(D(G(x,M)), fake) + \mathcal{L}_{BCE}(D(x,M), real), \tag{5.3}$$

where the real and fake are the labels that indicate if the image is real or synthesized by the generator (real is $(1, \ldots, 1)$, the fake is $(0, \ldots, 0)$). We can see that when the label of ground truth is real, and the predicted result closer to real will decrease the loss value, while the opposite increase the loss value. A similar situation when the label of ground truth is false. Combining the two losses, we can get the joint loss to train our generator:

$$\mathcal{L}_G = \mathcal{L}_{rec} + \alpha \mathcal{L}_{BCE}(D(G(x,M)), real), \tag{5.4}$$

where α is the weight to control the balance between reconstruction loss and adversarial loss. The adversarial loss for training the generator is different from that used to train the discriminator. Because for the discriminator, if it can correctly recognize the fake and real images, the loss should decrease. However, on the contrary, the generator that cannot fool the discriminator should get loss increased. We put the real label and the prediction of the fake images as the input of this loss function to achieve the expected results.

5.3 Datasets

As mentioned above, the latent space of the training dataset decides the generative model image synthesis capability. The generative inpainting model can only recover the images in which

Figure 5.3: Some indoor images from the dataset SUN360. Images courtesy of [52]

latent variables fit the latent space distribution. We decide to apply the model to perform the specific indoor scene inpainting. Because the omnidirectional datasets are limited, which can not provide so many different types of panoramas for training. And also

5.3.1 Real Indoor Omnidirectional Dataset

The dataset we built in Chapter 4 is too simple for the generative inpainting model to recover. So we collected indoor omnidirectional images from the SUN360 panorama dataset[52]. We only choose the images from the residence scene, which contains similar chairs, tables, and beds. So the distribution of the latent variables of these images is also limited in a small range, which can decrease the generative model's difficulty. Finally, we build the training set with 500 images, and the test set with 50 images. Some examples from the dataset are shown in Figure 5.3

5.4 Training Strategy

We train the GAN in the Alternating Training strategy following the protocol from [18], which is shown in Algorithm 1. This training strategy updates the discriminator and generator alternately, which means when update one of them, the parameters of the other one are fixed. Since we want the discriminator to compete with the generator, updating them together would lead the discriminator can not learn how to recognize the generator's flaws in the restored images. Similarly, the generator would try to follow moving constraints and would never converge.

To increase the stability of the GAN, [21] proposed to train the GAN in three phases. Phase I just use the reconstruction loss to train the generator and not use the discriminator for several epochs. Phase II fix the generator and just train the discriminator with adversarial loss function (Equation 5.3). Phase III train the network following the Algorithm 1 with adversarial loss function (Equation 5.3 for updating discriminator) and jointly loss function (Equation 5.4 for updating generator). We followed this strategy to train our generative inpainting network.

5.5 Results of Generative Inpainting

There is a pre-trained model of the original network, so we can first estimate the results by applying this pre-trained model on the omnidirectional images. The results are shown in Figure 5.4. We can observe that although the original network can recover the colors and textures of the images, it cannot recover the structure of objects well. The recovered lines in these

Algorithm 1: Alternating Training Strategy

Initialize the model parameters, optimizer, and build the data loader.
n is the current epoch, while the T_{train} is total epochs.
while $n < T_{train}$ **do**
    # Train Discriminator
    Compute the output of the generator;
    Put the generated data and the ground truth into Discriminator;
    Compute the GAN loss from the output of Discriminator;
    Compute the gradient of the weights of Discriminator by back-propagating the loss function;
    Update the weights of the Discriminator.
    # Train Generator
    Compute the output of the generator, and its reconstruct loss;
    Put the generated data and the ground truth into Discriminator;
    Compute the GAN loss from the output of Discriminator;
    Compute the gradient of the weights of Generator and Discriminator by back-propagating the total loss function;
    Only update the weights of the Generator.
end

images do not follow the correct direction of the omnidirectional way. This test proves that the conventional convolution layer cannot handle the omnidirectional images well.

We set the parameter α that controls the balance of the reconstruction loss and adversarial loss as 0.01. We trained the original model and adapted model 200 epochs for phase I, 80 epochs for phase II, and 400 epochs for phase III. The variation of the losses during the training process is shown in Figure 5.6. For different phases, although some losses have the divergent trend in the training procedure, finally they all converged to a constant value. And the variation of discrimination accuracies is shown in Figure 5.7. Since the discriminators were not used in phase I, there are only the accuracies results for phases II and III. In phase II, for both discriminators after several epochs training, they can recognize the fake and real inputs well. Then in phase III, they can still discriminate the real input well overall. But for the fake input, they are fooled by the generators.

For testing, following the strategy discussed in the previous chapter, we chose five areas of the image to perform the inpainting with both network's generators. The exact positions are shown in Figure 5.5. We randomly chose some inpainting results to present in the Figures 5.8, 5.9. We can see that in some places, the adapted generative model has better performance. For example, in the first column of the second group in Figure 5.8, the adapted model recovers the main structure of the ceiling, though the borders are still blurry. In contrast, the original model fails to recover anything. Also, in the fourth column of the first group in Figure 5.9, the adapted model recovers the main structure of the table, while the original one failed. Except for these places, we cannot tell too many differences from visual inspection.

From these intuitive results, we admit that the adapted generative model can improve the performance of omnidirectional inpainting, especially in the areas that are close to the poles. These areas have the most severe deformation in the panoramas. So the equirectangular convolutional operators can handle the non-uniform resolution of omnidirectional images.

Figure 5.4: The results of the inpainting for the omnidirectional images with the original generative model. For each group, the first row images are the original images while the second-row images are the inpainting results.

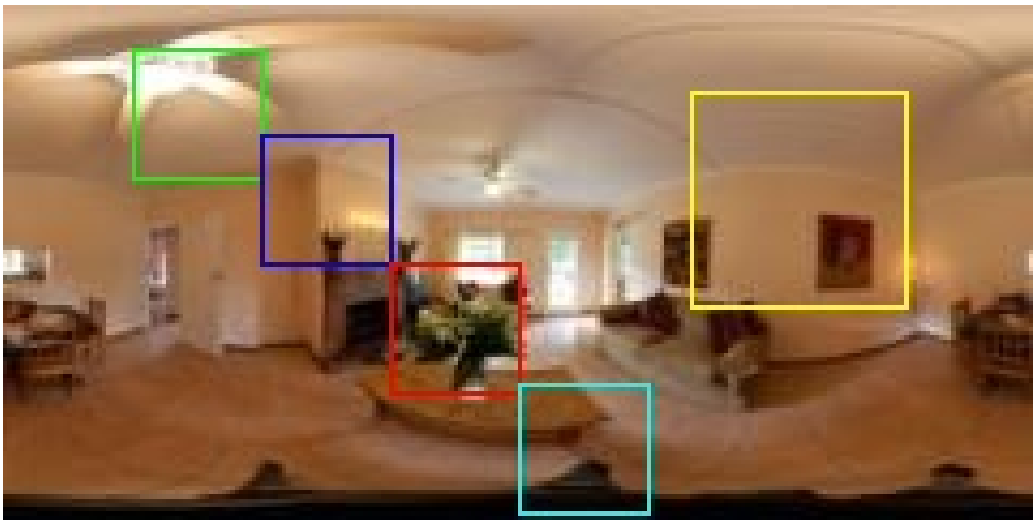


Figure 5.5: The five boxes indicate the inpainting areas in each image.

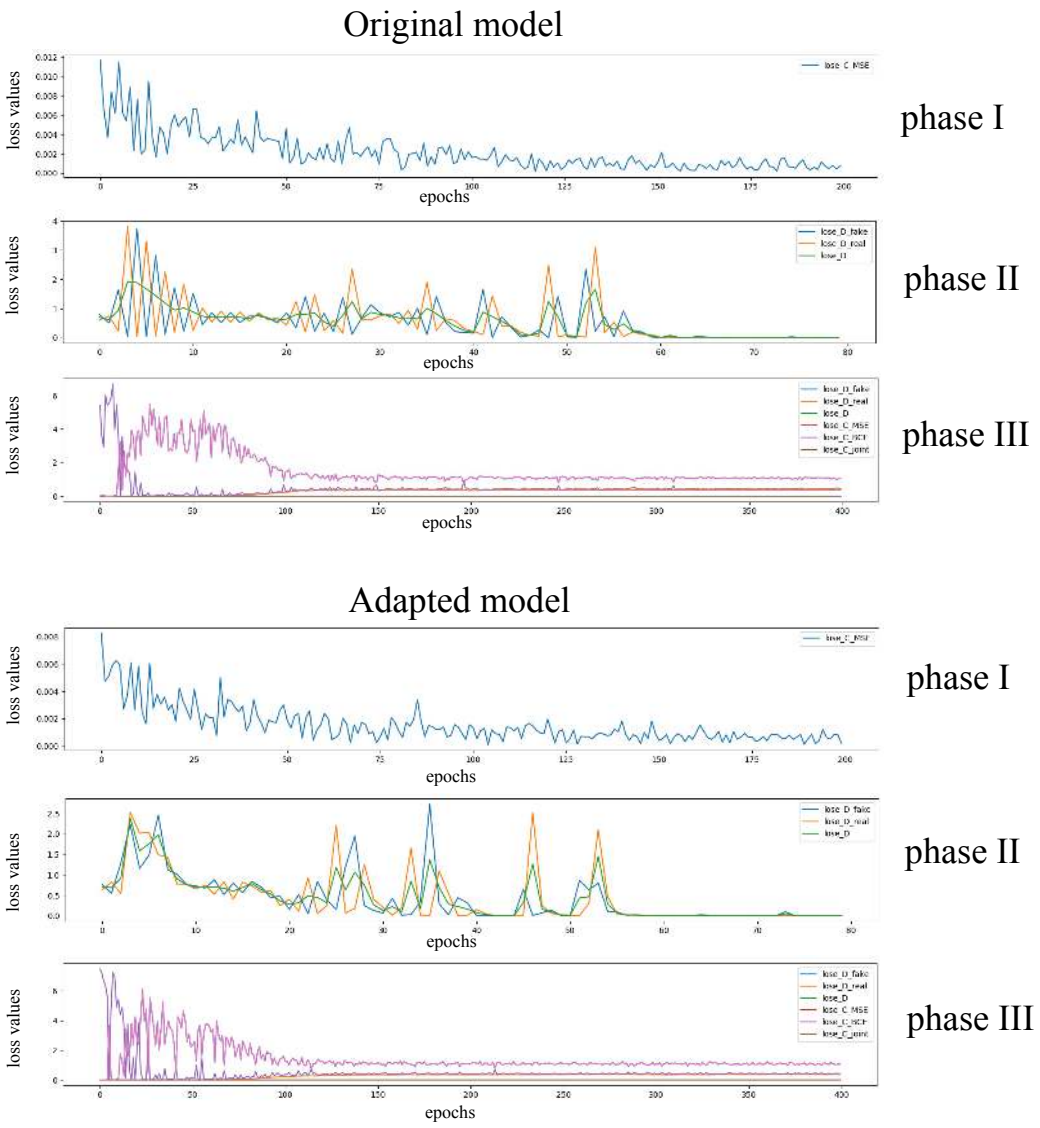


Figure 5.6: Training losses of the generative models. The upper three figures are the three training phases loss values of the original network, while the bottom three figures from the adapted network. The epochs of each phase are 200, 80, 400. For phase II and III, the loss_D is the adversarial loss of the discriminator while the loss_D_fake and loss_D_real are the components to compute the loss_D. And in phase III, loss_C_joint is the joint loss of generator while the loss_C_MSE and loss_C_BCE are the components to compute the loss_C_joint.

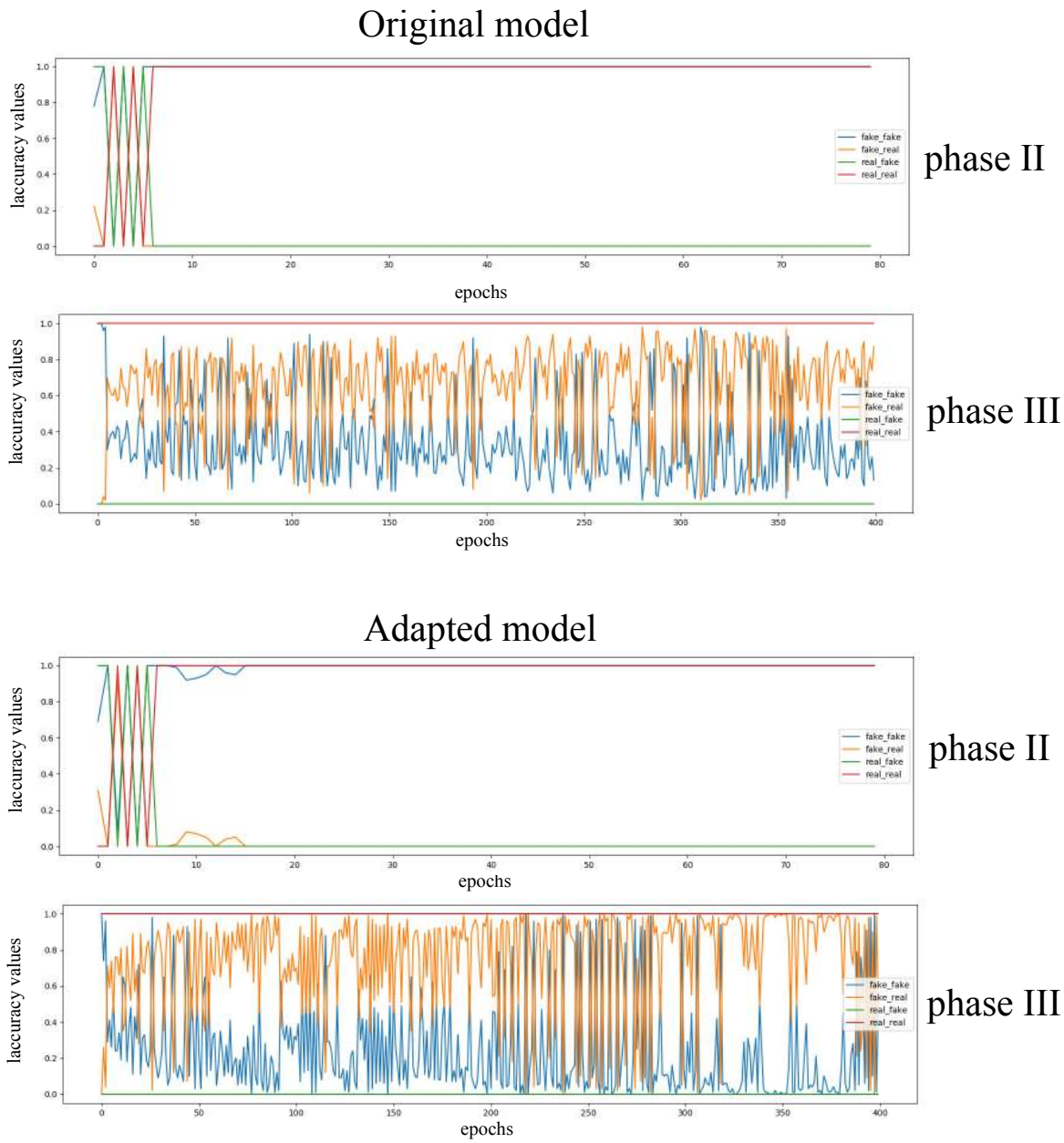


Figure 5.7: Accuracy results of the discriminators of the original inpainting network and the adapted one. For each image, the upper part is phase II accuracy while the bottom part is phase III accuracy. The fake_fake is the accuracy of the input being fake, and the discriminator considers it fake. While fake_real is the accuracy of that, the input is fake, and the discriminator considers it real. The other two accuracies have a similar meaning.

Figure 5.8: Part I: Inpainting with indoor images. Each column corresponds to one fixed missing region. For each group, the first row images are the input images. The second-row images are the original network inpainting results, while the last row images are the adapted network's inpainting results.

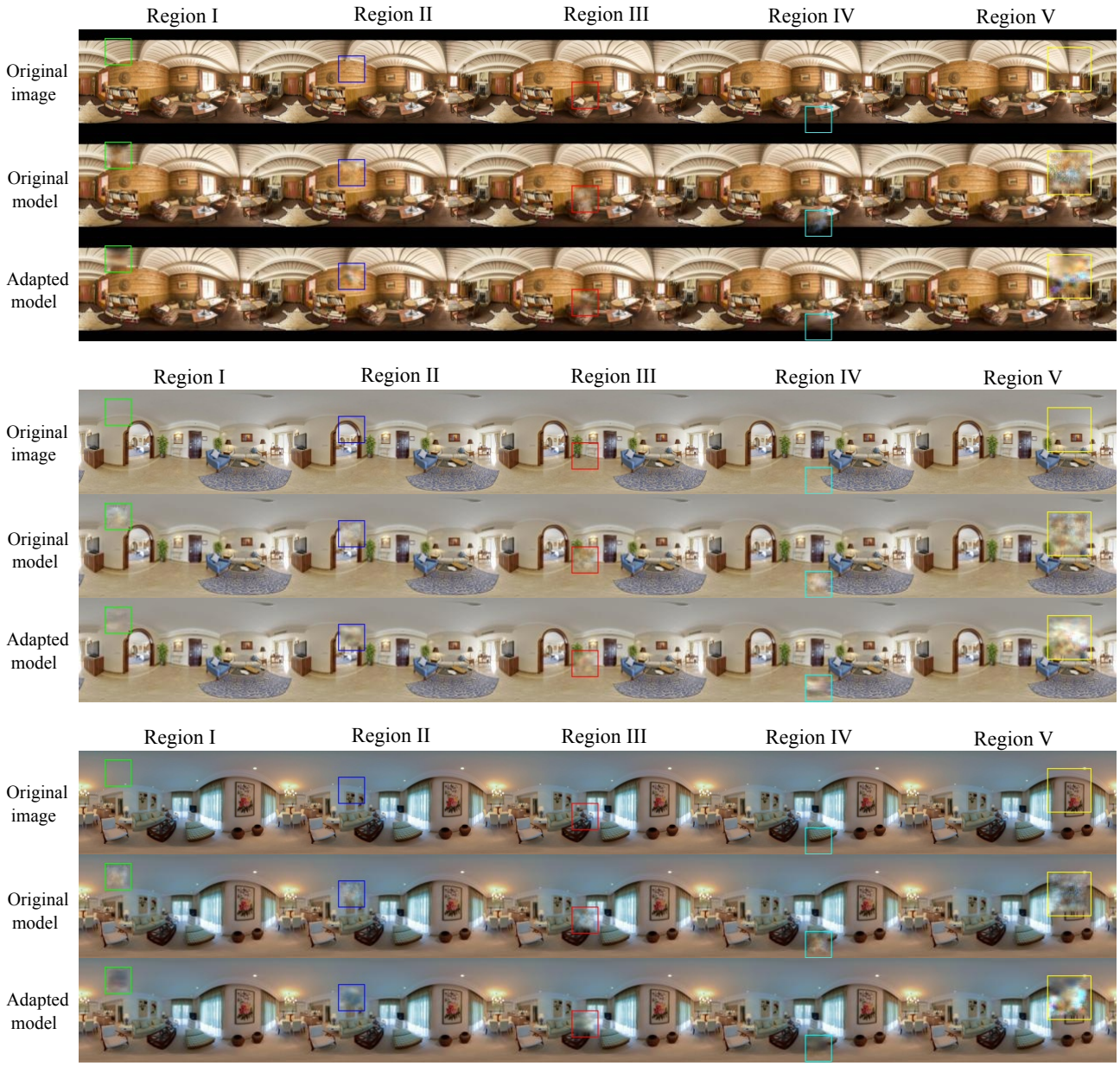


Figure 5.9: Part II: Inpainting with indoor images. Each column corresponds to one fixed missing region. For each group, the first row images are the input images. The second-row images are the original network inpainting results, while the last row images are the adapted network's inpainting results.

5.5.1 Evaluation

We employed MSE and LPIPS to evaluate the results. We computed the average values of each inpainting parts of all the images in the testing set. The results are shown in Table 5.1. In the regions I, IV, and V, the adapted generative model has the better performance than the original model with evaluation of the MSE and LPIPS. In the region II, the adapted model performs better within the LPIPS, while the original one performs better within the MSE. In the region III, the original model performs better within both the MSE and LPIPS. From the evaluation results, we find that in the five areas, the adapted inpainting network can perform better in the regions I, IV, and V. But the improvement is not significant among these areas. In the regions II and III, they show close performance. These evaluation results once again prove our previous observations. The equirectangular convolution can model the non-uniform

Table 5.1: Evaluation of inpainting results with MSE and LPIPS.

	MSE[1]		LPIPS [2]	
Region	Original	Adapted	Original	Adapted
I	0.0234	**0.0205**	0.1435	**0.1118**
II	**0.0203**	0.0225	0.1273	**0.1153**
III	**0.0132**	0.0141	**0.1038**	0.1107
IV	0.0194	**0.0188**	0.1156	**0.0934**
V	0.0398	**0.0329**	0.5394	**0.4755**
	[1] *Better closer to 0.*		[2] *Better closer to 0.*	

resolution of omnidirectional images better than the original one. However, the improvement is not significant notably from visual inspection.

Chapter 6

Conclusions

This master thesis proposed an adapted inpainting technique for omnidirectional images. For that, we first introduced the framework of large field-of-view camera models for representing omnidirectional images. This framework allows to map the images to a unified representation (assuming central projection) in the unit sphere, we can then manipulate and appropriately extract the information from omnidirectional images. We then investigated equirectangular convolution and spectral convolutional operators, which are the main components in the processing of the omnidirectional images by neural network models. The implementation of spectral convolution is the first challenge we faced since it involves the signal transformation between the spatial and frequency domains. Even applying optimized computing algorithms, spectral convolution is still much slower than the equirectangular convolutions. Based on these developments, we choose to build our adapted inpainting model using the projection equirectangular operators. To evaluate the modified models' performance, we proposed learning-based models to perform inpainting with a hierarchical bottom-up complexity: from simpler convolutional autoencoders to the more complex adversarial generative model. We constructed the models with the equirectangular operators and evaluated them to accomplish the reconstruction and inpainting on omnidirectional images. We also assessed the results of the works mentioned above with complimentary evaluation metrics such as mean squared error, structural similarity, and learned perceptual image patch similarity. These reconstructions and inpainting works provided a better understanding of the generative inpainting and provided some indications for tuning the generative inpainting network's structure and parameters.

Finally, we build our omnidirectional inpainting network with the state-of-the-art generative model. While designing and training this generative model, we faced the most challenging conditions during this project. Adversarial training requires setting carefully suited parameters, but no theoretical proof of convergence can be provided. We tuned this model parameters in the experiments, yet the setting of parameters found still produced inpainted images with some blur. We employed complementary metrics as perceptual and structural metrics to assess its performance. Although the improvement from visual inspection is limited, the evaluation results, from inpainting images from publicly available sequences, proved that the adapted generative model improved the inpainting performance.

Yet, the adapted network did not show enough improvements compared to the traditional model, which suggests to us as future work to evaluate deeper the training regime and the adapted losses. We also plan to design our models with spectral operators, which showed promising results in classification tasks compared to a classification model adapted with equirectangular projection operators. However, the model operators' complexity strongly increases, as well as the computational requirements when adopting spectral convolutions. We will then explore why the performance of the equirectangular operators is so different from the spectral ones.

Bibliography

[1] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera. Filling-in by joint interpolation of vector fields and gray levels. *IEEE transactions on image processing*, 10(8):1200–1211, 2001.

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009.

[3] T. Bülow. Multiscale image processing on the sphere. In *Joint Pattern Recognition Symposium*, pages 609–617. Springer, 2002.

[4] V. Caselles, A. Chambolle, and M. Novaga. Total variation in imaging. *Handbook of Mathematical Methods in Imaging*, pages 1455–1499, 2015.

[5] A. Chambolle, V. Caselles, D. Cremers, M. Novaga, and T. Pock. An introduction to total variation for image analysis. *Theoretical foundations and numerical methods for sparse recovery*, 9(263-340):227, 2010.

[6] T. F. Chan and J. Shen. Variational image inpainting. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 58(5):579–619, 2005.

[7] C. H. Choi, J. Ivanic, M. S. Gordon, and K. Ruedenberg. Rapid and stable determination of rotation matrices between spherical harmonics by direct recursion. *The Journal of Chemical Physics*, 111(19):8825–8831, 1999.

[8] T. Cohen, M. Geiger, J. Köhler, and M. Welling. Convolutional networks for spherical signals. *arXiv preprint arXiv:1709.04893*, 2017.

[9] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling. Spherical cnns. *arXiv preprint arXiv:1801.10130*, 2018.

[10] J. Courbon, Y. Mezouar, and P. Martinet. Evaluation of the unified model of the sphere for fisheye cameras in robotic applications. *Advanced Robotics*, 26(8-9):947–967, 2012.

[11] A. Criminisi, P. Pérez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on image processing*, 13(9):1200–1212, 2004.

[12] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen. Image melding: Combining inconsistent images using patch-based synthesis. *ACM Transactions on graphics (TOG)*, 31(4):1–10, 2012.

[13] C. Demonceaux, P. Vasseur, and Y. Fougerolle. Central catadioptric image processing with geodesic metric. *Image and Vision Computing*, 29(12):840–849, 2011.

[14] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1033–1038. IEEE, 1999.

[15] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[16] C. Geyer and K. Daniilidis. A unifying theory for central panoramic systems and practical implications. In *European conference on computer vision*, pages 445–461. Springer, 2000.

[17] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[19] H. Hadj-Abdelkader, E. Malis, and P. Rives. Spherical image processing for accurate visual odometry with omnidirectional cameras. 2008.

[20] J. Hays and A. A. Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics (TOG)*, 26(3):4, 2007.

[21] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)*, 36(4):1–14, 2017.

[22] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017.

[23] J. Ivanic and K. Ruedenberg. Rotation matrices for real spherical harmonics. direct determination by recursion. *The Journal of Physical Chemistry*, 100(15):6342–6347, 1996.

[24] J. Ivanic and K. Ruedenberg. Rotation matrices for real spherical harmonics. direct determination by recursion. *The Journal of Physical Chemistry A*, 102(45):9099–9100, 1998.

[25] F. Jacquey, F. Comby, and O. Strauss. Fuzzy edge detection for omnidirectional images. *Fuzzy sets and Systems*, 159(15):1991–2010, 2008.

[26] B. Khomutenko, G. Garcia, and P. Martinet. An enhanced unified camera model. *IEEE Robotics and Automation Letters*, 1(1):137–144, 2015.

[27] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[28] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566, 2016.

[29] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[30] A. Levin, A. Zomet, and Y. Weiss. Learning how to inpaint from global image statistics. In *null*, page 305. IEEE, 2003.

[31] H. Li, W. Luo, and J. Huang. Localization of diffusion-based inpainting in digital images. *IEEE Transactions on Information Forensics and Security*, 12(12):3050–3064, 2017.

[32] L. P. Matias, M. Sons, J. R. Souza, D. F. Wolf, and C. Stiller. Veigan: Vectorial inpainting generative adversarial network for depth maps object removal. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 310–316. IEEE, 2019.

[33] C. Mei and P. Rives. Single view point omnidirectional camera calibration from planar grids. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3945–3950. IEEE, 2007.

[34] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[35] K. A. Patwardhan, G. Sapiro, and M. Bertalmio. Video inpainting of occluding and occluded objects. In *IEEE International Conference on Image Processing 2005*, volume 2, pages II–69. IEEE, 2005.

[36] K. A. Patwardhan, G. Sapiro, and M. Bertalmío. Video inpainting under constrained camera motion. *IEEE Transactions on Image Processing*, 16(2):545–553, 2007.

[37] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Transactions on graphics (TOG)*, 22(3):313–318, 2003.

[38] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[39] Y. Ren, X. Yu, R. Zhang, T. H. Li, S. Liu, and G. Li. Structureflow: Image inpainting via structure-aware appearance flow. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 181–190, 2019.

[40] A. W. Robert Fisher, Simon Perkins and E. Wolfart. Sobel edge detector. https://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm, 2000.

[41] S. Saha. A comprehensive guide to convolutional neural networks — the eli5 way. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, 2018.

[42] D. Scaramuzza. Omnidirectional camera. *Computer Vision: A Reference Guide*, pages 552–560, 2014.

[43] D. Scaramuzza, A. Martinelli, and R. Siegwart. A flexible technique for accurate omnidirectional camera calibration and structure from motion. In *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, pages 45–45. IEEE, 2006.

[44] D. Scaramuzza, A. Martinelli, and R. Siegwart. A toolbox for easily calibrating omnidirectional cameras. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5695–5701. IEEE, 2006.

[45] K. Tateno, N. Navab, and F. Tombari. Distortion-aware convolutional filters for dense prediction in panoramic images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 707–722, 2018.

[46] E. Upenik, P. Akyazi, M. Tuzmen, and T. Ebrahimi. Inpainting in omnidirectional images for privacy protection. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2487–2491. IEEE, 2019.

[47] V. Usenko, N. Demmel, and D. Cremers. The double sphere camera model. In *2018 International Conference on 3D Vision (3DV)*, pages 552–560. IEEE, 2018.

[48] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[49] B. D. Wandelt and K. M. Gorski. Fast convolution on the sphere. *Physical review D*, 63(12):123002, 2001.

[50] B. D. Wandelt and K. M. Gorski. Fast convolution on the sphere. *Physical review D*, 63(12):123002, 2001.

[51] O. Whyte, J. Sivic, and A. Zisserman. Get out of my picture! internet-based inpainting. In *BMVC*, volume 2, page 5, 2009.

[52] J. Xiao, K. A. Ehinger, A. Oliva, and A. Torralba. Recognizing scene viewpoint using panoramic place representation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2695–2702. IEEE, 2012.

[53] R. Xu, X. Li, B. Zhou, and C. C. Loy. Deep flow-guided video inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3723–3732, 2019.

[54] X. Yan, J. Yang, K. Sohn, and H. Lee. Attribute2image: Conditional image generation from visual attributes. In *European Conference on Computer Vision*, pages 776–791. Springer, 2016.

[55] R. A. Yeh, C. Chen, T. Yian Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5485–5493, 2017.

[56] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5505–5514, 2018.

[57] A. Zalesny, V. Ferrari, G. Caenen, and L. Van Gool. Parallel composite texture synthesis. In *Texture 2002 workshop-ECCV*, 2002.

[58] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola. *Dive into Deep Learning*. 2020. https://d2l.ai.

[59] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[60] Z. Zhang, H. Rebecq, C. Forster, and D. Scaramuzza. Benefit of large field-of-view cameras for visual odometry. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 801–808. IEEE, 2016.

[61] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *European conference on computer vision*, pages 286–301. Springer, 2016.