# Image Interpolation and Denoising for Freeview Imaging Systems

Yu Mao

Doctor of Philosophy

Department of Informatics
School of Multidisciplinary Sciences
SOKENDAI (The Graduate University for Advanced Studies)

# Image Interpolation and Denoising for Freeview Imaging Systems

Yu Mao

A dissertation submitted to the Department of Informatics
School of Multidisciplinary Sciences
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at
SOKENDAI(The Graduate University for Advanced Studies)
September 2016

## Abstract

Using texture and depth maps of a single reference viewpoint, freeview imaging systems can synthesize a novel viewpoint image by depth-image-based rendering (DIBR)–translating texture pixels of the reference view to a virtual view, where synthesized pixel locations are derived from the associated depth pixel values. Browsing 3D scenes via a freeview imaging system creates a feeling of motion parallax, which is the most important source of 3D perception of human. Comparing to Computer Graphics technologies that can also provide 3D video contents, the advantage of freeview video is realness, which makes freeview video technology unreplaceable. However, the current freeview video system is far from satisfying.

In this thesis, via the means of solving the image interpolation and denoising problems with better quality via Graph Signal Processing(GSP), improvements are made to the freeview video system by enabling more interesting and attractive functionality, and yet reduce the application's energy consumption, making it affordable on modern smart devices. To accomplish the above achievements, we solved the problems from both the content consumer side and the content producer side. Especially, on the content consumer side, we improved the current system by proposing a $z$-dimensional movement extension. Lacking $z$-movements is causing users' unnatural feeling during watching. We made efforts to develop the $z$-dimensional mapping scheme and use GSP to solve the *expansion hole* problem – a patch of pixels sampled from an object surface in the reference view will be scattered to a larger spatial area, during a large $z$-movement into the scene. Second, on the content producer side, a depth image restoration scheme from sparse sensed samples is proposed to help save the energy of portable freeview video recording devices. Especially, we proposed quadtree decomposition and node-wise linear prediction model and the GSP-based residual restoration. The proposed image restoration algorithms in both applications outperformed the existing techniques in their specific area respectively. The $z$-dimensional image synthesized by the proposed scheme got up to 4.01 dB gain over a naive modification of VSRS 3.5, the current standard software of freeview video technology. And the reconstructed depth images by the proposal outperformed the results from the state-of-art LARK and BM3D techniques significantly by up to 1.26dB and 1.13dB respectively. Further, for the application of $z$-dimensional DIBR using sparse sensed depth image, the proposed schemes can work jointly and achieved up to 7.39 dB gain over traditional approach.

# Acknowledgments

It would not have been possible to write this thesis without the help and support of the kind people around me.

First, I'd like to thank Prof. Yusheng Ji for patiently giving me the support and guidance to archieve acedemic goals in the five year of study. And I'd like to thank Prof. Gene Cheung's for his tiredless effort in helping me with my research activities.

I also want to thank the other professors in my commitee, Prof. Noboru Sonehara, Prof. Imari Sato, Prof. Michihiro Koibuchi, Prof. Akira Kubota, for all the constructive advices and guidance I got from them.

I would also like to thank all members of Prof. Gene Gheung's lab and Prof. Yusheng Ji's lab. We often discuss research problems, give possible idea to each other and improve the qualities of our papers.

Last but not least, I would like to thank my family, and my friends, who have supported me throughout entire process.

# Contents

# List of Figures

# List of Tables

Table 1: Selected notations

| | |
|---|---|
| $f$ | focal length of the camera |
| $Q$ | voxel in the 3D space |
| $V_i, V_j$ | view points |
| $V_0, V_1, V_2$<br>$x, z_0, z_1, z_2$ | view points<br>$x-, z-$ coordinates of the view points |
| $P_0, P_1, P_2$ | projections of $Q$ on the image plane |
| $\alpha_0$ | horizontal distance between $Q$ and the primary axis of the camera |
| $\gamma_0$ | depth distance between $Q$ and the aperture of the camera |
| $(u_0, v_0), (u_1, v_1), (u_2, v_2)$ | coordinates of the projected pixels on the image plane |
| $\mu_G, \sigma$ | mean and standard deviation of Gaussian distribution |
| $\mu_L, l$ | mean and scale parameter of Laplace distribution |
| $p$ | target pixel to optimize |
| $\mathcal{R}$ | analysis window defined around $p$ |
| $S_w[p]$ | structural tensor on $p$ |
| $\Delta_x, \Delta_y$ | image gradients along $x-$ and $y-$ axes |
| $\mathbf{C}_p$ | coordinates of pixel p in the image |
| $\lambda_1, \lambda_2$ | eigen-values of the structural tensor |
| $a, b$ | long axis and short axis of the ellipse |
| $\rho$ | elongation factor to define the ellipse |
| $\phi$ | scaling factor to define the ellipse |
| $I_p$ | intensity value of pixel p |
| $\mathbf{s}$ | available pixels in the selecting ellipse |
| $\mathbf{s}^o$ | ground truth signal in the selecting ellipse |
| $\hat{\mathbf{s}}$ | signal reconstructed by our formulation |

# Chapter 1

# Introduction

## 1.1 Freeview Imaging System and Its Applications

The free-viewpoint video technology provides the functionality for users to interactively select a viewpoint of the 3D scene. It is an enhancement to multi-viewpoint video technology where the video scene from multiple viewpoints are captured, compressed and played back. In contrast to the limited available view points that is captured in a multi-viewpoint system, a free-viewpoint video system does not impose that the selected viewpoint corresponds to an existing camera viewpoint. In another word, beyond the restriction of showing an event only from the viewpoint of the cameras, freeview video technology allows a free navigation within the 3D video scene.

Freeview video systems relies on a partial 3D geometric description of the scene. The geometry of is typically described by a depth map, or disparity map, that specifies the per-pixel distance between a votex in the 3D space and the camera [1]. [1] Traditionally A depth image can be estimated from two images by identifying corresponding pixels in the multiple views, i.e., point-correspondences, that represent the same 3D scene point. Using this format, low-complexity DIBR view synthesis procedure such as 3D warping [2] can be used to create credible virtual view im-

[1] As there is a unique mutual translation between depth map and disparity map, we will not distinguish in the rest of this thesis.

ages, with the aid of inpainting algorithms to complete disocclusion holes [3–5]. While the conventional approach [6] transmits two (or more) pairs of color and depth maps from neighboring viewpoints for synthesis of an intermediate virtual view, recently, the authors in [7, 8] have shown that transmission of a single color-depth map pair can be more rate-distortion (RD) optimal, *if* the resulting larger disocclusion holes can be properly filled.

### 1.1.1 Applications

As the free-viewpoint video system provides the ability for users to interactively select any viewpoint at the same distance of the placed camera(s), it breaks the restriction of showing an event only from the viewpoint of the camera, but instead, allows a free angle changing within the 3D video Interesting applications include the selection of an arbitrary viewpoint for visualizing and analyzing sports or dynamic art (e.g., dance) actions. In such an application the head movement of the user can be captured by sensers and then the system will synthesize the video from the corresponding view point and present it to the user.

For details, considering the case of a live broadcast of a football match, it is often necessary for the audience to change view angle to see a player's movements which could be obstructed by another player. By rendering an appropriate viewpoint of the playing field, the player's movements can be well displayed in the virtual view. Free-viewpoint video technologies also simplify video training activities. For example, the training of dynamic activities such as martial arts or dancing can be simplified by allowing the trainee to select a viewpoint of the scene. Also, by deploying real-time freeview video streaming systems, we can realize telepresence, which is particularly useful in scenarios such as teleconferencing.

## 1.2 Motivation

The freeview video system is a promising application for entertainments. It is proposed to provide motion parallax – shifted viewpoint images of the scene for observation corresponds to the assumptive head motion of the user – in order to allow the user to form a depth percetion of the scene, thus a much richer and more interesting using experience. However, several shortcomings must be solved before it becomes a real option of the future form of entertainments.

First, $z$-dimensional movements is not supported. As described in Section 1.1, to create this motion parallax visual effect on a conventional 2D display, freeview imaging systems can capture images from different viewpoints of the same 3D scene using multiple closely spaced cameras. The user is then able to request views at a certain viewpiont to be rendered interactively on a 2D display. While the current DIBR can effiently synthesize image corresponding to the $x$-dimensional (left and right) viewpoint movements, it doesn't include solution for $z$-dimensional movements (forward and backward). The missing ability to respond to $z$-dimensional viewpoint movements will result unnatural experiences, especially in the scenarios where the request of the viewpionts are generated by the users' head movements. Second, the acquisition of the contents – video with 3D description of the scene – is hard and expensive. As freeview videos grant people the freedom to choose their own viewpoint to observe the contents, it is not likely that it will be shared by many people on a single device at the same time. This characteristic indicates that the future market of the technology would not be in the movie industry, where the expensive cost of creating the contents can be leveraged by the great amount of audiences who pay to watch it but will be in personal video sharing: each content is created and watched by individuals. Thus, making it easier to create freeview videos becomes a very important point in promote the application of the freeview video technology. One possible solution is to make the sensors work in sparse sensing mode, where less laser emission is required, resulting less energy consumption. A depth image interpolation operation will then be need to construct the full depth

image based on the sparse samples.

In this research, to overcome the above shortcomings of the freeview video technology. While the above two aspects comes from very different parts of the system, they share the same core problem of image interpolation and denoising. And we solved the problems by the same theory – GSP. Especially, to extend the current freeview image system to include $z$-dimensional movements, I first targeted on designing a $z$-dimensional image mapping scheme that is compatible with the existing standards and conventions on $z$-dimensional image synthesis, and then use GSP-based method to interpolate missing pixels. And for the second part I tried to use GSP theory to solve the problem of interpolation and denoising of sparsely sensed depth image, so sparse sensing mode can be enable on portable devices to save energy.

## 1.3 Contribution

During the research, I first proposed construction of novel $z$-dimensional DIBR-synthesized images in addition to the traditional $x$-dimensional DIBR applications. using the same popularized color-plus-depth representation, I propose to construct in addition novel images as observed from virtual viewpoints closer to the 3D scene, enabling a new dimension of view navigation. To construct this new image type, I first perform a new DIBR pixel-mapping for $z$-dimensional camera movement. I then identify expansion holes—a new kind of missing pixels unique in $z$-dimensional DIBR-mapped images—using a depth layering procedure. To fill expansion holes I formulate a patch-based maximum a posteriori (MAP) problem, where the patches are optimally spaced using diamond tiling. Leveraging on recent advances in graph signal processing (GSP), I define a graph-signal smoothness prior to regularize the inverse problem. Finally, I design a fast iterative reweighted least square (IRLS) algorithm to solve the posed problem efficiently. Experimental results show that our $z$-dimensional synthesized images outperform images rendered by a naïve modification of VSRS 3.5 by up to 4.01dB.

Further, to make the capture of freeview image / video possible via commodity devices energy-efficient, I tackle the technical challenge of fast restoration of depth images such as depth images given sparse, noisy pixel samples. I treat the depth images as a piecewise-smooth(PWS) image – an image with mostly smooth surfaces interrupted by discontinuities on the boundaries between foreground objects and background, and proposed an image model to serve the purpose of restoration. The processing operations that are performed locally on a pixel patch (thus amenable to parallel implementation) without the expensive cost of global search like nonlocal means (NLM), thus, is fast. Further, algorithms that require a variable number of iterations depending on characteristics of the current target pixel patch to satisfy an exit condition is also not desirable. With these stringent conditions, many state of the art image restoration algorithms become unsuitable for our application needs. Leveraging on recent advances in graph signal processing (GSP), I propose a joint denoising / interpolation scheme for PWS image restoration. I first pre-filter an image to obtain initial pixel estimates at each 2D grid pixel location. I then detect strong edges via spectral clustering I decompose an image into a quadtree representation, where each leaf represents a smooth image patch. Each patch is first coarsely approximated via 2D linear regression, then finely enhanced via a local graph - based filtering operation. Better image quality from our scheme is over LARK and BM3D is observed via experiments.

## 1.4 Organization

The outline of the thesis is as follows. I first present the background in Chapter 2. The $z$-dimensional image reconstruction via DIBR is presented in Chapter 3. Then the joint denoising and interpolation for sparse sensed depth images are presented in Chapter 4. And finally, the summary and future work are presented in Chapter 5.

# Chapter 2

# Background

## 2.1 Interactive Freeview Video Streaming System and its Applications

Free viewpoint Image system offers compelling interactive experience by allowing users to switch to any viewing angle. Freeview Image System consists of three components: A camera array that captures the scene, a server that calculates and encodes the captured images into pre-designed representations and a client that interacts with the server and reconstructs the images. An illustration of conventional Freeview Image System is shown in Fig. 2-1.

As in conventional free viewpoint TV systems [9], we assume that a 1D array

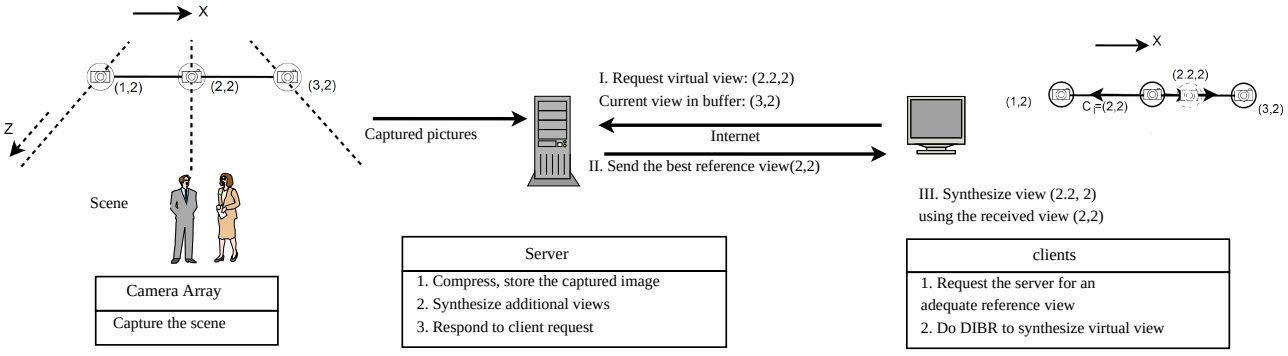



Figure 2-1: Interactive freeview streaming system and the example of how the system present the requested virtual view, $V_j = (2.2, 1)$, $V_i = (2, 2)$

of closely spaced cameras along the horizontal $x$-axis is used to capture a 3D scene from different viewpoints.

Then the representation of the 3D scene is computed on the server based on the captured images, such as 3D mesh, sprites, color-plus-depth, etc. Among all the possible representations, we focus on the color-plus-depth representation [1, 10], as it is adaptive to any different scenarios and has a bounded complexity in synthesize the images to be presented to the user. The color-plus-depth representation consists of several color-depth image pairs, where the depth image contains the per-pixel depth information, i.e. the distance between the camera and the pixel in the 3D space.

The depth images can be computed via a stereo-matching algorithm [11] based on the captured images. Alternatively, depth sensors like Microsoft Kinect or Intel Realsense can capture both color and depth images simultaneously. Then both color and depth images are compressed and stored at a server using standard coding tools like MVC and 3D-HEVC [12, 13].

### 2.1.1 Scene Capturing

In this section we discuss the acquisition of the depth map. In a freeview system, multiple cameras capture the same scene. To further allow free-viewpoint synthesis, besides the texture images captured by the cameras, a signal is needed to describe the geometric structure of the 3D scene as previously discussed. In the texture + depth representation, the depth of a color pixel can be calculated via stereo matching–estimating the pixel-correspondences across the captured multi-view images. By assigning a depth value to each pixel of the captured texture image and combining those depth values into an image, a depth image is created. The calculation of depth by estimating stereo matching is an difficult task in many situations. For example, the pixel value of a votex in the 3D space can vary across the captured images due to the change of illumination across the views or the differences in the camera settings. The signal ambiguity while identifying the

potentially corresponding points. This results in an unreliable identification of the point-correspondences and thus inaccurate depth values. For the purpose of research, relatively accurate depth images had been captured in laboratories using expensive professional cameras with carefully designed illustration. However, it is not realistic for daily usage.

On the other hand, recent technology advances has brought to us the affordable active depth sensor such Microsoft Kinect and Intel Realsense. These sensors consist of an infrared laser projector combined with a monochrome CMOS sensor, which captures video data in 3D under any ambient light conditions. Nevertheless, the actively sensed depth images also suffer from the limited accuracy and can hardly be directly used for the purpose of creating freeview video. Further, though portable version of depth sensor are now available on some mobile devices, their usability is restricted by the considerable energy consumption.

We now formulate the image capturing process using the popular *pinhole camera* model [14].

**Pinhole camera model**

The optical system can be simplified as a *pinhole camera* model, shown in Fig. 2-2. In the figure, the camera's *aperture* is denoted by $V_0$ and serves as the center of the camera system. The *focal length*—the distance between the aperture and the *image plane*—is denoted by $f$. The image plane is orthogonal to the system's *principal axis*, which indicates the camera's viewing direction and goes through $V_0$.

A 3D coordinate system is established using $V_0$ as its origin. $\gamma_0$ is the "depth" distance between a voxel in the 3D scene and $V_0$—a projected component onto the principal axis. $\alpha_0$ is the horizontal distance between the voxel and the principal axis. Given this coordinate system, in Fig. 2-2 a voxel $Q$ in the 3D space with a particular color intensity is projected through $V_0$ onto the image plane as pixel $P_0$. Note that the image projected by a pinhole camera is flipped upside down and left to right; in the sequel we revert this inversion to present the captured image more naturally. An example of the inverted captured image is shown in Fig. 2-3(a).

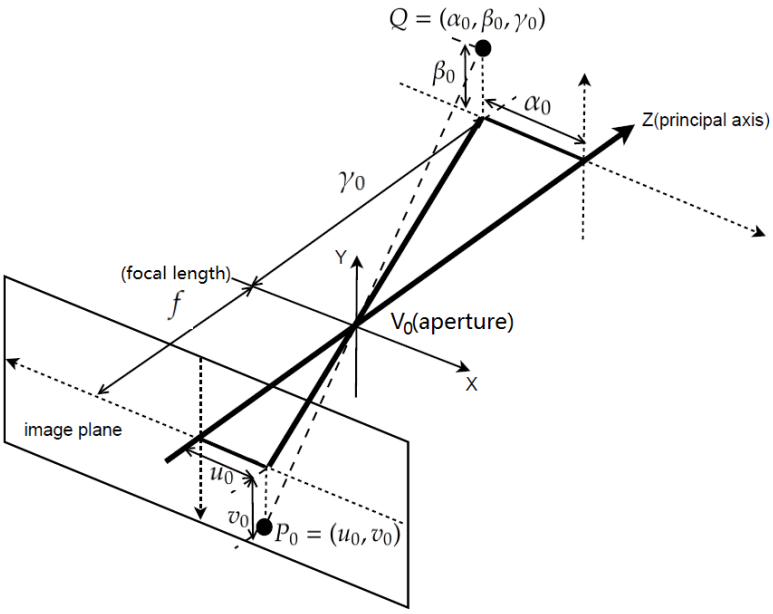



Figure 2-2: Projection of a 3D voxel onto an image plane using a pinhole camera model.

In the inverted image plane, pixel $P_0$'s 2D coordinates in the captured image are denoted by $(u_0, v_0)$, where $u_0$ can be computed using similar triangles:

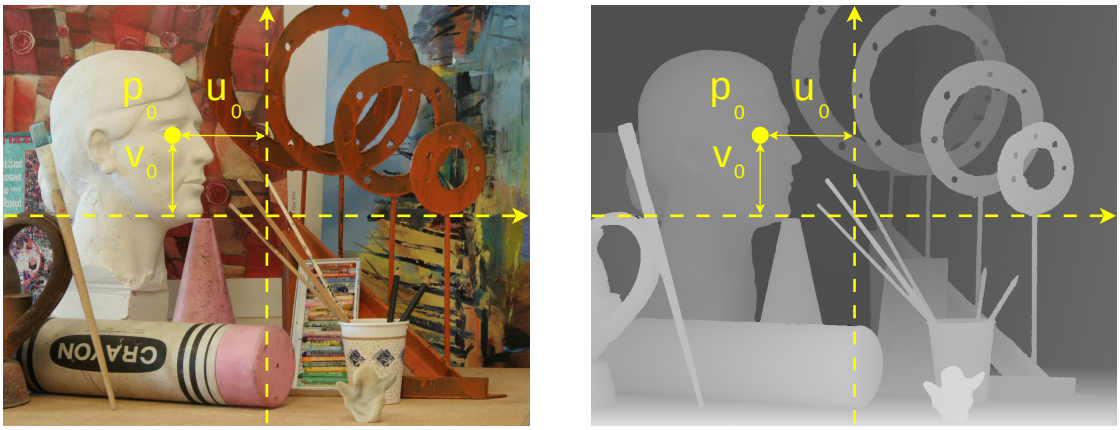
$$u_0 = \alpha_0 \frac{f}{\gamma_0} \tag{2.1}$$

The vertical coordinate $v_0$ can be computed using the same procedure.

The described mapping procedure from voxels in 3D space to pixels on a 2D image plane retains color information as observed from a particular camera viewpoint. The same procedure can also be used to capture a depth image, where each 3D voxel $Q$ now reflects its depth value $\gamma_0$ (or its reciprocal, *disparity*, $1/\gamma_0$). Fig. 2-3(b) shows a captured depth image from the same viewpoint.

### 2.1.2 Interactive View Switching

In interactive freeview video streaming system [15, 16], the scene will be browsed via conventional 2D display, so there exists an asymmetry between data available at the server and data consumption at the client. Specifically, while the server contains




(a) texture image (b) depth image

Figure 2-3: Texture / depth image pair from the same camera viewpoint.

coded 3D data that can synthesize a large number of views, a client can observe only one viewpoint of the 3D scene at a time rendered on a 2D display. Thus, the client will periodically request new views for observation, and in response the server must transmit appropriate data for rendering of the requested views.

More specifically, in a conventional interactive freeview video streaming system [15, 16], given a set of color-plus-depth image pairs at camera-captured viewpoints $\mathcal{X} = \{1, \ldots, N\}$ at some fixed $z$-coordinate $z_0$ relative to the 3D scene, a client can select any viewpoint $x$ between leftmost view 1 and rightmost view $N$ to observe the scene. A virtual view $x$, $1 < x < N$ and $x \notin \mathbb{Z}^+$, can be synthesized using color and depth image pairs of the nearest left and right camera-captured views $\lfloor x \rfloor$ and $\lceil x \rceil$ via DIBR [6] (to be discussed in detail). Thus, when a client requests virtual view $x$, color and depth map pairs at one or both camera-captured viewpoints $\lfloor x \rfloor$ and $\lceil x \rceil$ (an RD decision based on bandwidth cost and synthesized view quality [7, 8]) will be transmitted to the client. On the other hand, client side collect the view switch commands from the user. The view switch requests can be either input from conventional input devices such as keyboards and mouses, or generated by analyzing the users head and eye movements, which can be recorded by various devices, from a simple RGB camera to very sophisticated eye trackers.

### 2.1.3 View Synthesis

We now discuss Depth Image Based Rendering(DIBR)–the conventional image synthesis process corresponding to the color-plus-image representation. DIBR takes one or more color-depth images pairs as input and then map the pixels in the color image into the image to be synthesized. The new coordinates of the pixels is calculated via geometrics based on the movement of the view point, the depth value of the pixel and a set of camera parameter discussed in 2.1.1.

#### Pixel Mapping by DIBR

We formulate the conventional DIBR process, *i.e.*, synthesis of a new horizontally shifted viewpoint image given texture and depth map pair of the original captured camera view. In Fig. 2-4(a), we show the top view of the previously described optical system, allowing us to focus on the geometric relationships among objects on the $x$-$z$ plane. A virtual camera with aperture $V_1 = (x_0 - \Delta x, z_0)$ is located at distance $\Delta x$ from the reference camera with aperture $V_0$, resulting in a corresponding shift in the principal axis. The location of the image plane remains the same, though the center also moves from $V_0$ to $V_1$. The previously projected pixel $P_0 = (u_0, v_0)$ in the old image plane can now be translated to a location $P_1$ in the new image plane as follows. First, pixel $P_0$ is back-projected to position $Q$ in the 3D space, and then is re-projected to location $P_1 = (u_1, v_0)$ in the new image plane. Again using similar triangles, we can calculate $u_1$:

$$u_1 = \alpha_1 \frac{f}{\gamma_0} = \alpha_0 \frac{f}{\gamma_0} + (\alpha_1 - \alpha_0)\frac{f}{\gamma_0} = u_0 - \Delta x \frac{f}{\gamma_0} \tag{2.2}$$

Thus, given the $x$-dimensional camera movement by $\Delta x$, the new horizontal coordinate $u_1$ can be computed using $f$ and $\gamma_0$. Further, for $x$-dimensional camera movement there is no change in the vertical coordinate. The original and synthesized images after $x$-dimensional camera movement are shown in Fig. 2-4(b) and (c) respectively, where the pixels of the sculpture's eye are highlighted. Note that multiple pixels from the old image plane with different depth values may be

mapped to the same pixel location in the new image. In this case, we keep the pixel with the shallowest depth, as foreground objects occlude background objects.

#### Missing Pixels after Mapping

In Fig. 2-4(c), we can find *holes* in the synthesized virtual view, *i.e.*, a pixel location in the virtual view that has no corresponding pixel in the reference view. There are three kinds of holes. The first is *disocclusion holes*, which are spatial locations that are occluded by foreground object(s) in the reference view, but become exposed in the virtual view. Disocclusion holes are large continuous areas appearing between foreground and background. There have been many depth-based image inpainting techniques proposed to complete disocclusion holes [3–5].

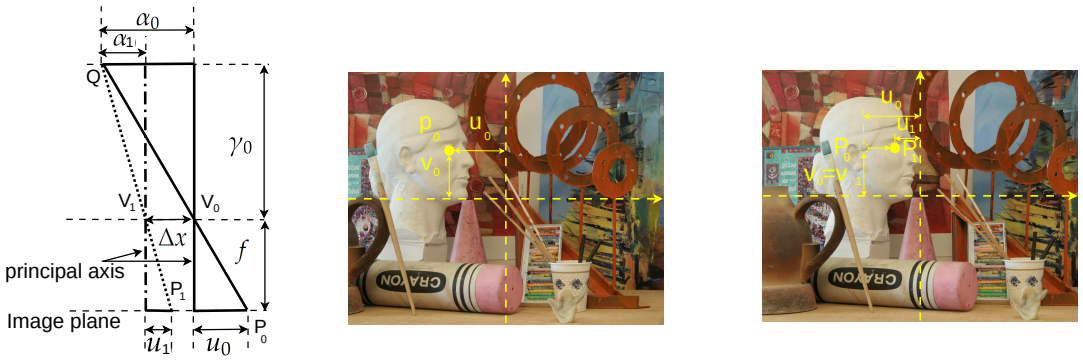
The second kind is *out-of-view* holes, which appear at the left or right boundaries of the image. They exist because the image's field of view has changed due to the $x$-dimensional camera movement. The out-of-view holes are filled using the same method as disocclusion holes.

The last kind of holes, known as *rounding holes*, appear when an object covers a slightly larger spatial area in the synthesized image due to the change of viewpoints. This change in size is typically very small during $x$-dimensional camera movement, and rounding holes are filled using simple local interpolation methods, such as bilinear interpolation [6].

Finally, we note that in practice a pixel $(u_0, v_0)$ in the reference view is copied to the *nearest integer position* to $(u_1, v_0)$, *i.e.* $(u_0 + \text{round}(\Delta u), v_0)$, on the 2D image grid of the virtual view. This rounding process introduces errors in the synthesized view. We will study the error due to this rounding process in a later section.

### 2.1.4 System Analysis

In this section, we discuss the evaluation of free viewpoint video systems.




(a) movement (b) reference (c) synthesized

Figure 2-4: DIBR for $x$-dimensional camera movement.

### Quality of the Synthesized Images

As a video system, the most important evaluation metric of free viewpoint video system is the quality of output images. The objective evaluation is performed by comparing the images synthesized at a certain point $A$ using the information of the texture and structure from another point $B$ with the ground truth image taken by a camera at point $A$. Better synthesis algorithm will synthesize the images more similar to the ground truth.

### Valid Viewpoint Domain

Another evaluation metrics of free viewpoint video system is the possible range of viewpoints. The idea of free viewpoint video system comes from the multi-viewpoint video systems, where the structural information of the scene is not available, thus, only provides the images captured by the cameras. In another word, multi-viewpoint video system cannot provide continuous view switch, making it more functional than entertaining. By exploiting the structural information, free viewpiont system can provide smooth view switches, which allows more interesting applications such as telepresence/virtual reality by synthesize the images corresponding to the user's head movements to be implemented. However, the conventional free viewpiont video system is still far from satisfying as it can only generate images from the view points at the distance to the scene, making it unable to respond to all the complex head movements. In this research, the problem is solve

as the system proposed provides the functionality of synthesizing $z$-dimensional images, thus can respond to any head movements.

**Scene-Capturing Component**

Despite the client side, the performance of the scene-capturing component should also be considered. Capturing freeview videos used to be very expensive and exhausting, as the expensive professional-level cameras need to be carefully placed in pre-designed positions, which is required by the stereo matching algorithms. Recently, the depth sensors on portable devices brought to us by the latest technology advances freed us from the camera arrays, providing a much more usable scene-capturing method for freeview videos. However, the depth sensors cost a lot of energy as their actively emit lasers when working. On the other hand, the battery capacity has been a problem for smart portable devices ever since their invention and there seems no solution that can be reached in the near future. It is predictable that the energy cost will become the bottleneck of these devices. In this research, we tried to solve the problem by providing a depth interpolation scheme which can recover the depth image from sparsely sensed samples – less samples means less active sensing, thus, less energy consumption.

**Other Aspects**

Another main component of the system is the server, which compresses and stores the captured information and respond the requests from the client side. There is rich literature on the compression [17–20] and streaming [21–23] of freeview/multiview videos. As this research doesn't focus on this part, corresponding discussions are not provided here.

## 2.2 Image Interpolation and Denoising

### 2.2.1 Inverse Imaging Problems

Inverse Imaging Problems are a set of problems that aims at reconstructing the image from limited samples and have been a fundamental and yet active topic in the field of signal processing. The samples are often noisy and contain incomplete information about the target parameter or data due to physical limitations of the measurement devices. In the literature, the signal degrading process are expressed by the following equation:

$$\mathbf{Y_s} = \mathbf{S}(\mathbf{H} * \mathbf{Z}) + \mathbf{N_s} \tag{2.3}$$

, where $\mathbf{Z}$ is the original signal, its convolution with the blurring matrix $\mathbf{H}$ are sampled by the sampling grid $\mathbf{S}$, further the samples $\mathbf{Y_s}$ often contains some additive noise, which is denoted by $\mathbf{N_s}$. Inverse problems are then aimed at the reconstruction of original signal $\mathbf{Z}$ from the observation $\mathbf{Y_s}$. A inverse problem following the above generic formulation involves too much uncertainty and is hard to solve, thus people often focus on one specific aspect of the model to make the problem clear and solvable, which divides the inverse problems into three categories: interpolation, deblurring and denoising. People can apply the methods from the above categories sequentially to solve practical problems. However, having these categories does not mean that the problems have to be studied separately. In the contrast, we are never short of examples of research that involve two or even more factors. And in this chapter, we tackle the joint interpolation and denoising problem for a special category of images – the depth images.

### 2.2.2 Related Works

There are many works targeting at solving the image interpolation and/or denoising. For example, kernel regression are a set of versatile methods, that can solve joint interpolation and denoising problems. Especially recent research about adaptive kernel regression LARK [24], has gained much fame for its ability to reconstruct

the image from very limited noisy samples. The problem we are trying to solve is similar to that in the LARK paper, despite that we focus on a more However, LARK has some restrictions that is hard to overcome. First, it adopts an iterative processing progress on the signal without a clear terminating rule, and is thus potentially slow. Second, its performance on edge or conner area is not satifying. In the specific problem of reconstructing the depth maps from sparse and noisy samples, we are able to develop solution for the joint task of denosing and interpolation without leveraging on iterations, and thus is faster. Further, the image reconstruction performance, especially on edge area are improved. There

### 2.2.3 MAP Formulation

The image restoring techniques employed in thesis follows the maximum a posteriori probability (MAP) formulation. MAP method is a natural extension of Fisher's method of maximum likelihood (ML), from a Bayesian perspective.

With the distribution of the observations $s$ known as $f(s|s_o)$ with the ground truth signal denoted by $s_o$ as a parameter, ML estimations predict the most possible signal $\hat{s_o}$ by maximizing the likelihood function:

$$\hat{s}_{ML} = argmax_{s_o} f(s|s_o) \tag{2.4}$$

However, as the image restring problems are often ill-posed, ML estimation cannot yield satisfying results. To solve the problem, we need to assume a prior distribution $p(s_o)$ of $s_o$. Then the posterior probability of $s$ is given as $posterior(s) = f(s|s_o)p(s_o)$ by Bayesian's theorem, thus the MAP estimation is given by:

$$\hat{s}_{MAP} = argmax_{s_o} f(s|s_o)p(s_o) \tag{2.5}$$

MAP employs an augmented optimization objective which incorporates a prior distribution over the quantity one wants to estimate and can therefore be seen as a regularization of ML estimation. Obviously, the prior probability function $p(s_o)$

plays as a key factor of the estimation. We will discuss the prior used in this research in the next section.

### 2.2.4 Graph-based Image Processing

In this research the prior used in the MAP estimation is derived using GSP. GSP is the study of signals that live on structured data kernels described by graphs [25], leveraging on spectral graph theory [26] for frequency analysis of graph-signals. Graph-signal priors have been derived for inverse problems such as denoising [27–29], interpolation [30], bit-depth enhancement [31] and de-quantization [32]. The common idea among these works is the assumption that the desired graph-signal is smooth with respect to a properly chosen graph $\mathcal{G}$ that reflects the structure of the signal. Typically one assumes that the appropriate graph $\mathcal{G}$ is known *a priori* as available side information, or can be discovered from noisy and / or partial observations of the signal. In this work, we assume the latter case and construct a suitable graph $\mathcal{G}$ from available DIBR-synthesized pixels for joint denoising / interpolation of pixels in a target patch.

#### Graph Laplacian Smoothness Prior

We now introduce the Graph Laplacian Smoothness Prior. First, we overview the basic concepts in graph signal processing (GSP) and graph spectral analysis [25]. As done in [25], we will focus on undirected graphs with non-negative edge weights. A weighted undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ consists of a finite set of vertices $\mathcal{V}$ with cardinality $|\mathcal{V}| = N$, a set of edges $\mathcal{E}$ connecting vertices, and a weighted adjacency matrix $\mathbf{W}$. $\mathbf{W}$ is a real $N \times N$ symmetric matrix, where $W_{i,j} \geq 0$ is the weight assigned to the edge $(i, j)$ connecting vertices $i$ and $j$, $i \neq j$. $W_{i,j} = 0$ means vertices $i$ and $j$ are not connected, *i.e.*, $(i, j) \notin \mathcal{E}$.

Given a defined graph $\mathcal{G}$, the *degree matrix* $\mathbf{D}$ is a diagonal matrix whose $i$-th diagonal element is the sum of all elements in the $i$-th row of $\mathbf{W}$, *i.e.*, $D_{i,i} = \Sigma_{j=1}^{N} W_{i,j}$.

The *combinatorial graph Laplacian* $\mathbf{L}$ (graph Laplacian for short) is then:

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \tag{2.6}$$

Because $\mathbf{L}$ is a real symmetric matrix, there exists a set of eigenvectors $\phi_i$ with corresponding real eigenvalues $\lambda_i$ that decompose $\mathbf{L}$, *i.e.*,

$$\mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T = \sum_i \lambda_i \phi_i \phi_i^T = \mathbf{L} \tag{2.7}$$

where $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalues $\lambda_i$ ordered from smallest to largest on its diagonal, and $\mathbf{\Phi}$ is an eigenvector matrix with corresponding eigenvectors $\phi_i$ as its columns. It can be shown that $\mathbf{L}$ is positive semi-definite [25], *i.e.* $\mathbf{x}^T\mathbf{L}\mathbf{x} \geq 0$, $\forall \mathbf{x} \in \mathbb{R}^N$, which implies that the eigenvalues are non-negative, *i.e.* $\lambda_i \geq 0$. The eigenvalues can be interpreted as frequencies of the graph; thus using $\mathbf{\Phi}^T$ as a transform, any input graph-signal $\mathbf{x}$ can be decomposed into its graph frequency components via $\mathbf{\Phi}^T\mathbf{x}$, where $\alpha_i = \phi_i^T\mathbf{x}$ is the $i$-th frequency coefficient. $\mathbf{\Phi}^T$ is called the *graph Fourier transform* (GFT) [33].

The expression $\mathbf{x}^T\mathbf{L}\mathbf{x}$—called the *graph Laplacian regularizer* [29]—captures the total variation of the signal $\mathbf{x}$ with the respect to the graph $\mathcal{G}$ [34]:

$$\mathbf{x}^T\mathbf{L}\mathbf{x} = \frac{1}{2}\sum_{(i,j)\in\mathcal{E}} W_{i,j}\,(x_i - x_j)^2. \tag{2.8}$$

In words, $\mathbf{x}^T\mathbf{L}\mathbf{x}$ is small if connected nodes $x_i$ and $x_j$ have similar values for each edge $(i, j) \in \mathcal{E}$, *or* if the weight $W_{i,j}$ is small.

$\mathbf{x}^T\mathbf{L}\mathbf{x}$ can alternatively be expressed in terms of graph frequencies $\lambda_i$:

$$\mathbf{x}^T\mathbf{L}\mathbf{x} = \sum_i \lambda_i \alpha_i^2 \tag{2.9}$$

Thus a small $\mathbf{x}^T\mathbf{L}\mathbf{x}$ means that the energy of signal $\mathbf{x}$ is mostly concentrated in the low graph frequencies—*smooth* with respect to the defined graph. We will employ the Laplacian regularizer $\mathbf{x}^T\mathbf{L}\mathbf{x}$—*i.e.*, the desired signal is mostly smooth with respect to a defined graph—as our image prior in the subsequent section.

### 2.2.5 Metrics for image reconstruction quality

Digital image reconstruction quality assessment – how similar is the ground truth and the reconstructed image from partial and/or noisy samples – has been an active topic ever since the beginning of the study of digital image processing. Ideally, image reconstruction quality should be evaluated by human viewers, as the images are aimed to be browsed by human eyes. however, it is not feasible to have viewers to evaluate all the images, given the tremendous number of images and the slow speed of human assessment. Numerous metrics has been proposed to automate the image assessment procedure. Among the metrics, Peak signal-to-noise ratio (PSNR) is most straight forward and yet most widely used to evaluate the image reconstruction. PSNR is most easily defined via the mean squared error (MSE). Given a $m \times n$ ground truth image $I$ pixels and its reconstruction $K$, MSE is defined as:

$$MSE(I, K) = \frac{1}{mn} \sum_{i=1}^{n} \sum_{j+1}^{m} (I(i, j) - K(i, j))^2 \tag{2.10}$$

and the PSNR(in dB) is defined as:

$$PSNR(I, K) = 10 \log_{10}(\frac{255^2}{MSE(I, K)}) = 20 \log_{10} 255 - 10 \log_{10} MSE(I, K) \tag{2.11}$$

as we can see from the definition, PSNR and MSE come from the idea of calculating the absolute difference between two images. Despite PSNR's popularity, it has also received many critics as it sometimes be inconsistent with human perception. Lots of effort has been paid to develop human-perception-consistent metrics. And the Structural Similarity(SSIM) [35] index is among the most successful proposals. It considers image degradation as perceived change in structural information, while also incorporating important perceptual phenomena, including both luminance masking and contrast masking terms. Structural information is the idea that the pixels have strong inter-dependencies especially when they are spatially close. These dependencies carry important information about the structure of the

objects in the visual scene. Luminance masking is a phenomenon whereby image distortions (in this context) tend to be less visible in bright regions, while contrast masking is a phenomenon whereby distortions become less visible where there is significant activity or "texture" in the image.

The SSIM value is calculated on various windows of an image. The measure between $w(I)$ and $w(K)$ – $I$ and $K$ with the window $w$– is as following:

$$SSIM(w(I), w(K)) = \frac{(2\mu_{w(I)}\mu_{w(K)} + c_1)(2\sigma_{w(I)w(K)} + c_2)}{(\mu^2_{w(I)} + \mu^2_{w(K)} + c_2)(\sigma^2_{w(I)} + \sigma^2_{w(K)} + c_2)} \tag{2.12}$$

where $\mu_{w(I)}$ and $\mu_{w(K)}$ are the averages of the two windows, $\sigma_{w(I)}$ and $sigma_{w(K)}$ are the variances and $\sigma_{w(I)w(K)}$ is the covariance. $c_1$ and $c_2$ are division stabillizer. The SSIM between $K$ and $I$ is then calculated by averaging over the windows.

There are other metrics that adapts to special scenarios such as *3DSwIM* for DIBR synthesized images and some more metrics that has finer models of human visual perception. However, the detailed discussion is beyond the scope of this thesis.

# Chapter 3

# The Construction of z-dimensional DIBR-synthesized Images

## 3.1 Introduction

The promise of free viewpoint video [9] is to provide users the freedom to choose any vantage point from which to construct a viewpoint image for observation of a 3D scene. To enable free viewpoint, a conventional multiview imaging system contains an array of horizontally spaced cameras to capture color maps (conventional RGB images) and depth maps (per-pixel distance between objects in the 3D scene and the capturing camera) from different viewpoints—a format called *color-plus-depth* [1]. The captured images are subsequently encoded at the sender using standardized multiview coding tools such as 3D-HEVC [13].

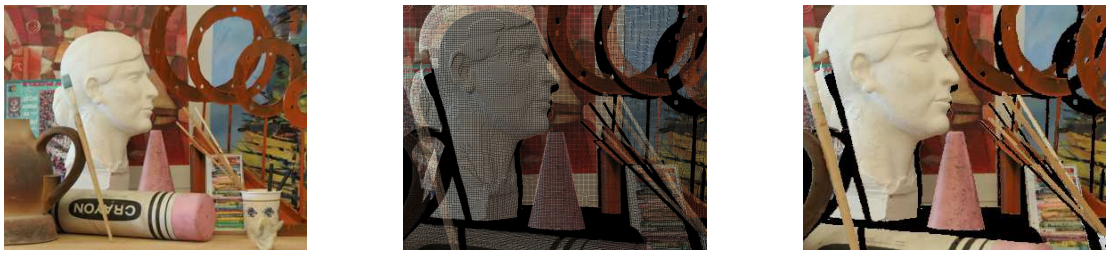
At the receiver, a novel image from a virtual viewpoint—a horizontally shifted camera angle from the captured views—can be synthesized using *depth-image-based rendering* (DIBR) techniques such as 3D warping [6]. In a nutshell, DIBR maps each color pixel in a reference view to a 2D grid location in the virtual view, using disparity information provided by the corresponding depth pixel. Due to *occlusion* (visible spatial areas in the virtual view that are occluded by foreground objects in the reference view), there are missing pixels in the virtual view called *disocclusion*

*holes*. They are subsequently completed using inpainting algorithms designed specifically for disocclusion hole filling [3–5]. For small camera movement from reference to virtual view along the $x$-dimension (camera moving left or right), this DIBR view synthesis plus inpainting approach works reasonably well [9], and is the conventional approach in the free view synthesis literature.

In immersive applications such as teleconferencing, a sitting viewer observes rendered images on a 2D display, where the image viewpoints are interactively adjusted according to the tracked locations of the viewer's head as he shifts left or right [36]. The resulting *motion parallax* effect can greatly enhance the viewer's depth perception in the 3D scene [37]. To enable this interactive view navigation in streaming systems over networks, previous works have optimized strategies for coding of color and depth maps [15, 16], error resilience [38], packet scheduling [39, 40], caching [41] and reference view selection [42] for visual quality and/or view interactivity.

Besides $x$-dimensional head movement (moving one's head left or right), $z$-dimensional head movement (moving one's head front or back) is also very natural for a sitting observer. However, while interactive streaming for $x$-dimensional view navigation has been investigated extensively [15, 16, 38–42], to the best of our knowledge, *the problem of synthesizing viewpoint images corresponding to large $z$-dimensional virtual camera movements using the color-plus-depth format has not been formally studied from a classical image interpolation perspective*. We address this problem formally in this chapter, extending the capabilities of previous interactive free-viewpoint systems.

When the virtual camera is located closer to the 3D scene than the reference view camera, objects close to the camera will increase in size in the virtual view. This means that the aforementioned pixel-to-pixel mapping during DIBR from reference to virtual view cannot complete entire surfaces of rendered objects, resulting in *expansion holes* [43]. Note that expansion holes differ fundamentally from disocclusion holes in that the objects are visible in the reference view(s), but *the pixel sampling in the reference view is not sufficient for rendering in the synthesized view*




(a) captured far view (b) DIBR-synthesized (c) expansion holes filled

Figure 3-1: Examples of disocclusion and expansion holes: a) camera-captured color map; b) $z$-dimensional DIBR-synthesized view, where *disocclusion holes* are larger contiguous empty regions next to foreground object boundaries, and *expansion holes* are smaller empty regions on the surfaces of foreground objects; c) synthesized view with expansion holes filled by our proposed scheme.

*when z-directional camera movement is significant*. See Fig. 3-1 for an illustration of expansion and disocclusion holes.

In this chapter, we propose a methodology to construct $z$-dimensional DIBR-synthesized images, including an efficient solution to the challenging expansion hole filling problem. We first perform a new DIBR pixel-mapping for $z$-dimensional camera movement. We then identify disocclusion holes using a depth layering procedure. To fill expansion holes, we formulate a patch-based maximum a posteriori (MAP) problem, where the patches are optimally spaced using diamond tiling. Leveraging on recent advances in graph signal processing (GSP) [25], we define a graph-signal smoothness prior to regularize the inverse problem. Finally, we design a fast iterative reweighted least square (IRLS) algorithm [44] to solve the posed problem efficiently. Experimental results show that our $z$-dimensional synthesized images outperform images rendered by a naïve modification of VSRS 3.5 by up to 4.01dB in PSNR and noticeably in two other quality metrics, SSIM [35] and 3DSwIM [45]. We claim that we are the first to rigorously formulate the expansion hole filling problem in DIBR images as a MAP problem, and provide a graph-based interpolation algorithm that solves it efficiently.

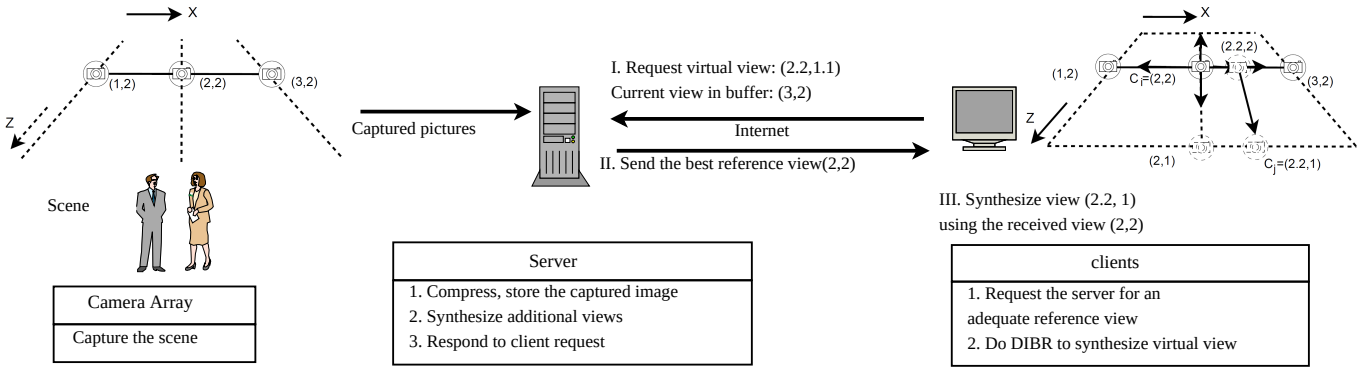



Figure 3-2: Interactive free view streaming system and the example of how the system present the requested virtual view, $V_j = (2.2, 1), V_i = (2, 2)$

## 3.2 DIBR for $z$-dimensional movement

In contrast to the conventional systems that restrict virtual cameras placed at the same distance with the captured cameras, in our enhanced IMVS system a client can in addition synthesize images at virtual camera location $(x, z)$, where the $z$-coordinate can differ from the captured cameras'—*i.e.* $z \neq z_0$—as shown in Fig. 3-2. A virtual camera location $(x, z_1)$ for small $z_1$ means that the viewpoint is closer to the 3D scene than camera location $(x, z_0)$, $z_1 < z_0$. It means that objects close to the camera now appear bigger, and the field-of-view is narrower; see Fig. 3-1(c) for an example. *Note that in general one cannot practically set up a 2D camera array along both x- and z-dimension; the physical camera closer to the scene along the z-dimension will be visible and obstruct the view of a camera further from the scene.* Thus, given only a conventional 1D camera array setup, a client must synthesize a novel image at virtual viewpoint $(x, z_1)$ using *only* color and depth map pairs at camera-captured views $(\lfloor x \rfloor, z_0)$ and / or $(\lceil x \rceil, z_0)$. As shown in Fig. 3-1(b), this results in expansion holes that require filling; the formulation and algorithm for expansion hole filling are discussed in Section 3.6 and 3.7 respectively.

Next, we overview the $z$-dimensional mapping.

### 3.2.1 $z$-dimensional Mapping

In this section we extend DIBR to enable $z$-dimensional camera movement also. The top view of the optical system is shown in Fig. 3-3(a), where the virtual camera with aperture $V_2 = (x_0, z_0 - \Delta z)$ is shifted from reference camera with aperture $V_0$ by $\Delta z$ along the principal axis. The image plane is shifted correspondingly while the principal axis remains the same. Thus, when a pixel $(u_0, v_0)$ on the old image plane is translated to the new image plane, we can compute the new horizontal coordinate $u_2$ again using similar triangles:

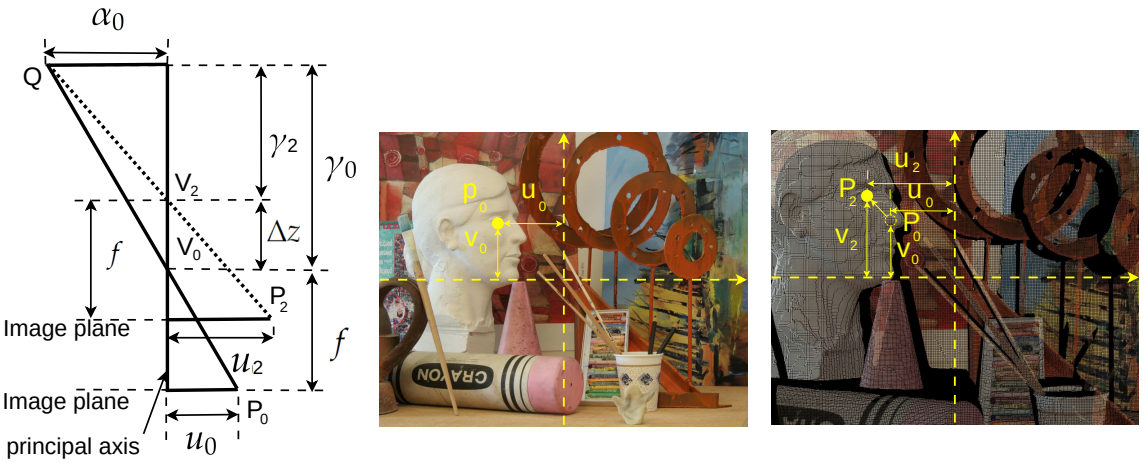
$$u_2 = \alpha_0 \frac{f}{\gamma_2} = u_0 \frac{\gamma_0}{\gamma_2} = u_0 \frac{\gamma_0}{\gamma_0 - \Delta z} \tag{3.1}$$

Unlike the $x$-dimensional camera movement, the vertical coordinate of a pixel also changes during a $z$-dimensional camera movement. The new vertical coordinate can be calculated similarly as follows:

$$v_2 = v_0 \frac{\gamma_0}{\gamma_0 - \Delta z} \tag{3.2}$$

The original and synthesized images after the $z$-dimensional camera movement towards the 3D scene are shown in Fig. 3-3(b) and (c) respectively, with example pixels highlighted. Besides holes we encountered during a $x$-dimensional camera movement, we now have a scattering of missing pixels over the surfaces of objects. We call this new kind of holes—unique for $z$-dimensional camera movement—*expansion holes*. Though in principle the expansion holes are similar to rounding holes, they are much larger in size, especially when there is a large $z$-dimensional movement, and hence the interpolation quality has an important influence on the overall synthesized image perception. We will describe a scheme specifically for expansion hole filling in a later section. Note that though we focus on the $z$-directional camera movement in this work, in practice when any mixture of $x$-/$y$-/$z$-directional camera movements is possible, we can take the following strategy. First, we differentiate between disocclusion holes and expansion holes. Second,

we determine if the amount of $z$-directional movement is larger than a threshold. If so, our algorithm can be used for interpolation of expansion holes. If not, a local hole filling strategy as done in VSRS software is employed.




(a) movement (b) reference (c) synthesized

Figure 3-3: DIBR for $z$-dimensional camera movement.

### 3.2.2 Likelihood – Rounding Noise in DIBR-mapped pixels

As previously discussed, when we compute new coordinates for pixels mapped from the reference to virtual view, the computed quantities are in general not integers and must be rounded to the nearest 2D grid positions for display, resulting in errors we call *rounding noise*. *To the best of our knowledge, the extent and characteristics of rounding noise in the DIBR pixel-to-pixel mapping procedure have not been studied systematically in the literature before.* We characterize such rounding noise in this section empirically.

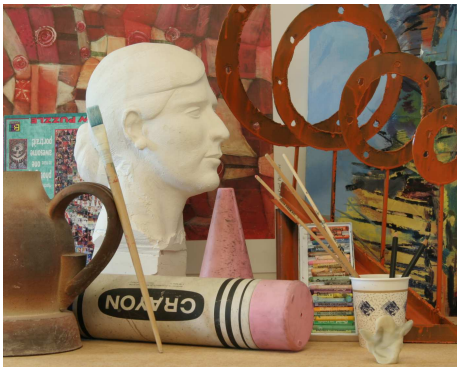
We first seek a suitable statistical description for rounding noise. Specifically, we compute the differences between DIBR-mapped pixels from reference views and camera-captured ground truth pixels of the same views to construct an error distribution. We then describe the resulting distribution using two popular noise models—Gaussian and Laplacian distribution with probability density functions

(PDF):

$$f_{Gaussian}(x|\mu_G, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(x-\mu_G)^2}{2\sigma^2}\right\} \tag{3.3}$$

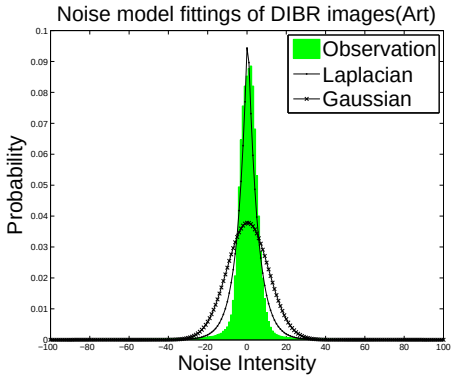
$$f_{Laplacian}(x|\mu_L, l) = \frac{1}{2l} \exp\left\{-\frac{|x-\mu_L|}{l}\right\} \tag{3.4}$$

where $\mu_G$ and $\mu_L$ are the means, and $\sigma^2$ and $2l^2$ are the variances of the distributions respectively. The reason we restrict our attention to only Gaussian and Laplacian distributions is because the subsequent MAP problem formulations stemming from these models become relatively straight-forward $l_2$ and $l_1$-norm minimization, solvable via efficient iterative algorithms. Problem formulation and corresponding optimization algorithms are described in Section 3.6 and 3.7 respectively.

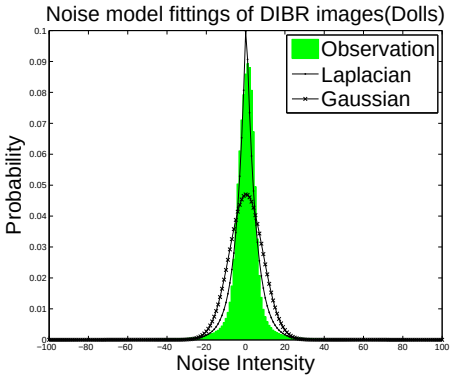


(a) `Art` sequence

(b) `Dolls` sequence




(c) DIBR noise distribution and best fitted error models for `Art`




(d) DIBR noise distribution and best fitted error models for `Dolls`

Figure 3-4: Fitting noise models to DIBR noise distribution for `Art` and `Dolls`

Using squared error as a criterion, we compute the error-minimizing model parameters for the Gaussian and Laplacian distributions separately. The best fitted models are shown in Fig. 3-4 for image sequences `Art` and `Dolls`. We observe

that the root mean square errors (RMS) of the Laplacian model (0.0008 for `Art` and 0.0006 for `Dolls`) are smaller than that of Gaussian model (0.0020 for `Art` and 0.0017 for `Dolls`). Thus, we conclude that the Laplacian distribution is a better statistical description of rounding noise, and is the preferred noise model for later optimization.

Having chosen a rounding noise model and identified the expansion holes, we next denoise DIBR-mapped pixels and complete expansion holes via a unified *maximize a posteriori* (MAP) formulation for a given target patch with center at pixel $p$. Specifically, first we divide the image into individual patches with overlaps, where each patch contains similar pixels (Section 3.4). To restore pixels in each patch, we introduce a graph-signal smoothness prior to regularize an otherwise under-determined problem (Section 3.5). Finally, we formulate the MAP estimation problem in Section 3.6.

## 3.3 Expansion Holes

We now describe two different kinds of missing pixels in a $z$-dimensional DIBR-synthesized image, each requiring a different filling method and how we distinguish them. For simplicity, we only describe the DIBR view synthesis procedure using a single reference color and depth map pair in the following discussions.

### 3.3.1 Definition

Before we can identify the expansion holes, we need to first properly define them. For intuition, we examine a magnified piece of a $z$-dimensional DIBR-synthesized image in Fig. 3-5, where disocclusion holes are enclosed by yellow lines and expansion holes are enclosed by blue or green lines. Formally, we define an expansion hole as follows: a spatial area of an object's surface in the virtual view, whose corresponding area in the reference view is visible but smaller in size. Since the amount of expansion is typically evenly distributed over the surface of an object,

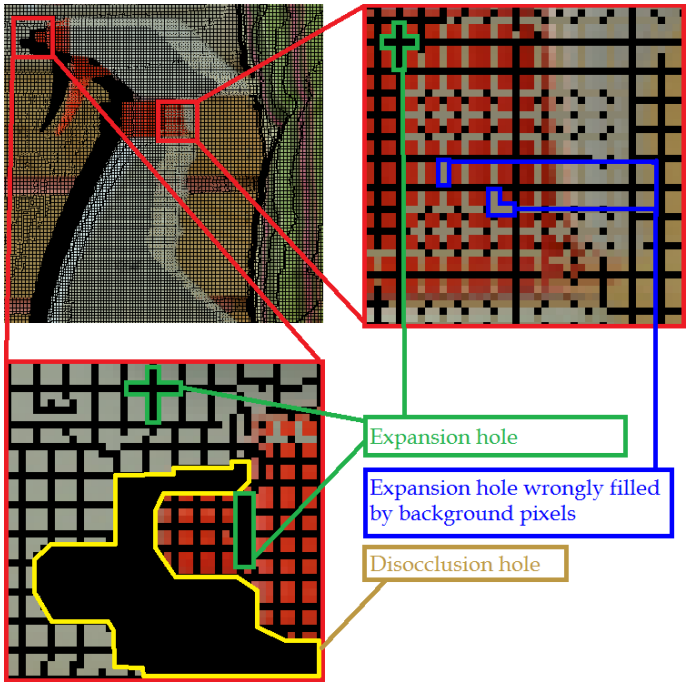


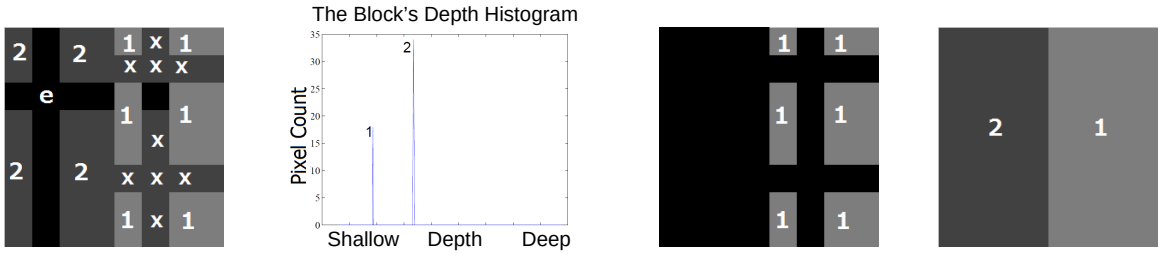

Figure 3-5: Expansion holes and disocclusion holes in the output image of DIBR

expansion holes are small “cracks" that spread all over the surface of an enlarged object. As observed, this signal characteristic is very different from the large and continuous disocclusion holes.

Further, while a disocclusion hole always appears empty, *expansion holes can often be wrongly filled with background pixels during DIBR*. The reason is because the size expansion for a foreground object surface is typically larger than a background surface during a $z$-dimensional camera movement, and hence in an expansion hole due to enlargement of foreground object surface, a background pixel can be mapped from reference view erroneously. Fig. 3-5 shows an example with a foreground detergent bottle in front of a background wooden panel. However, the pixels of the panel erroneously fill the expansion holes of the bottle. Hence it is important to correctly identify and remove these erroneously mapped background pixels *before* performing completion of expansion holes. We describe our proposed procedure next.

### 3.3.2 Identification

To identify pixels from the same physical object in the DIBR synthesized view, we adopt a *depth layering* approach. Specifically, for a given pixel block in the synthesized view, we first construct a histogram containing depth values of pixels in the block. Fig. 3-6(b) shows an example. Peaks in the histogram are labeled as layers ordered from shallow depth to deep depth. Fig. 3-6(a) shows the depth pixels in the block with assigned layer numbers.




(a) block (b) depth histogram (c) first layer (d) reconstructed

Figure 3-6: Examples of depth layers and corresponding histogram: a) pixels in a depth block are classified into depth layers and empty pixels; b) corresponding histogram of depth values for the block; c)the first depth layer separated from the second depth layer; d) reconstructed depth block.

The identification is performed layer-by-layer, starting from the shallowest, so that when interpolating missing pixels in layer $a$, each pixel in layer $b > a$ that is inside a convex set spanned by pixels in layer $a$ is treated as an empty pixel. In Fig. 3-6(c), we shown that layer 2 pixels are treated as empty pixels during expansion hole filling of layer 1. The reconstructed depth block is shown in Fig. 3-6(d). Having identified available pixels in a depth layer for interpolation of empty pixels, we next formulate our joint denoising / interpolation problem for expansion hole filling.

## 3.4 Adaptive Kernel

We first divide the pixels in the same depth layer into overlapping patches. In particular, we adaptively select patch shapes based on observed signal characteristics,

because the same physical object can have distinct *textural patterns* that influence how pixels should be interpolated. For example, if the captured object is a red and blue striped shirt, then pixels inside a blue stripe should be interpolated using only neighboring blue pixels.

### 3.4.1 Patch Selection via Adaptive Kernel

To select the first pixel patch for joint denoising / interpolation, we first select a pixel $p$ in the depth layer (*e.g.*, the top-left pixel), and calculate an *adaptive kernel* centered at $p$, similarly done in [24] (we discuss our implementation difference from [24] in details later). There are two basic steps. In the first step, the *principal gradient* in a local neighborhood is derived via computation of the *structure tensor* [46]. The structure tensor $S_w(p)$ defined on pixel $p$'s location $\mathbf{C}_p$ can be computed as:

$$S_w(p) = \begin{bmatrix} \sum_{r\in\mathcal{R}} w(\mathbf{C}_r)(\Delta_x(\mathbf{C}_p,\mathbf{C}_r))^2 & \sum_{r\in\mathcal{R}} w(\mathbf{C}_r)\Delta_x(\mathbf{C}_p,\mathbf{C}_r)\Delta_y(\mathbf{C}_p,\mathbf{C}_r) \\ \sum_{r\in\mathcal{R}} w(\mathbf{C}_r)\Delta_x(\mathbf{C}_p,\mathbf{C}_r)\Delta_y(\mathbf{C}_p,\mathbf{C}_r) & \sum_{r\in\mathcal{R}} w(\mathbf{C}_r)(\Delta_y(\mathbf{C}_p,\mathbf{C}_r))^2 \end{bmatrix} \quad (3.5)$$

where $\mathcal{R}$ defines a square neighborhood around pixel $p$, $\Delta_x(\mathbf{C}_p,\mathbf{C}_r)$ and $\Delta_y(\mathbf{C}_p,\mathbf{C}_r)$ are the color image gradients[1] along the $x$- and $y$-axis at pixel $p$ respectively, and $w(\mathbf{C}_r)$ is a weight assigned to neighbor $r$. Weights are determined by a Gaussian kernel, which is normalized so that $\sum_{r\in\mathcal{R}} w(\mathbf{C}_r) = 1$.

Having computed $S_w(p)$, we perform eigen-decomposition on the matrix, which summarizes the local gradients within $\mathcal{R}$. The ratio between the larger $\lambda_2$ and smaller eigenvalue $\lambda_1$ indicates relative strength of the principal gradient in the patch $\mathcal{R}$, and the eigenvector $v_2$ corresponding to the larger eigenvalue $\lambda_2$ indicates the direction of $\mathcal{R}$'s principal gradient. For example, in the case that a strong edge

[1] Gradient $\Delta(\mathbf{C}_p,\mathbf{C}_r)$ at pixel $p$ is computed as the intensity difference from a neighbor $r$ divided by the distance between $p$ and $r$.

is present in the pixel patch, the principal gradient will be *orthogonal* to the edge. See Fig. 3-7 for an illustration.

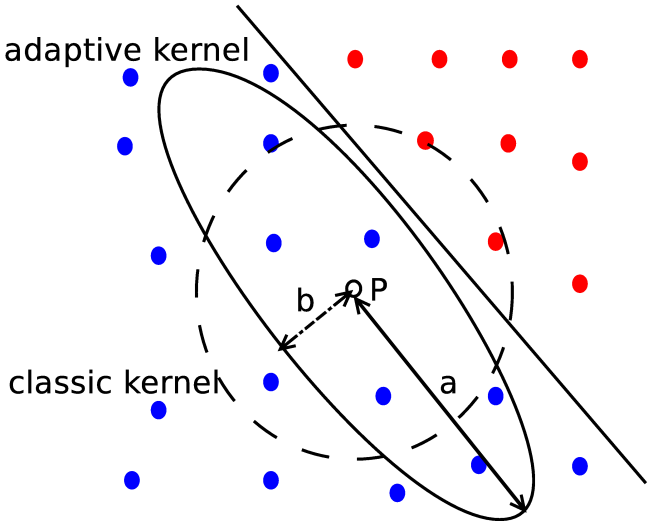



Figure 3-7: Illustration of patch selection via adaptive kernel. An ellipse is elongated perpendicular to the principal gradient, so that similar pixels are selected for pixel interpolation.

Finally, an ellipse centered at the target pixel $p$ is defined to identify a subset of pixels in the same depth layer for joint denoising / interpolation. The ellipse has minor axis aligned with the tensor eigenvector $v_2$, and the major axis orthogonal to the minor axis. In particular, let $a$ and $b$ be the major and minor radius, *i.e.*, in the rotated coordinate system $(x', y')$,

$$\left(\frac{x'}{a}\right)^2 + \left(\frac{y'}{b}\right)^2 = 1 \tag{3.6}$$

We compute $a$ and $b$ as:

$$a = \rho\,\phi, \quad b = \rho^{-1}\,\phi \tag{3.7}$$

$\rho = \sqrt{\frac{\lambda_2+\omega}{\lambda_1+\omega}}$ is the *elongation factor*. $\rho$ reflects the relative strength of the principal gradient, with $\omega > 0$ for numerical stability. Using $\rho$, the shape of the kernel is kept circular in flat areas, where $\lambda_1 \approx \lambda_2 \approx 0$, and elongated when near a strong edge ($\lambda_2 \gg \lambda_1$). The idea is to construct an ellipse elongated perpendicular to the principal gradient of the patch, so we can include enough similar pixels for joint denoising / interpolation.

$\phi = m\sqrt{\lambda_1\lambda_2 + \epsilon}$ is a *scaling factor*, where $m$ is a size parameter and $\epsilon$ is used for numerical stability. Since $\sqrt{\lambda_1\lambda_2}$ is the geometric mean of the tensor's eigenvalues, $\phi$ induces a large kernel in a flat area to average over more pixels (better denoising), and induces a smaller kernel in a heavily textured area (avoid blurring).

In Fig. 3-7(a), an example ellipse is elongated to contain only blue neighboring pixels, resulting in a texture-adaptive pixel kernel. In contrast, a classic kernel will be a circle with a fixed radius, containing both blue and red pixels.

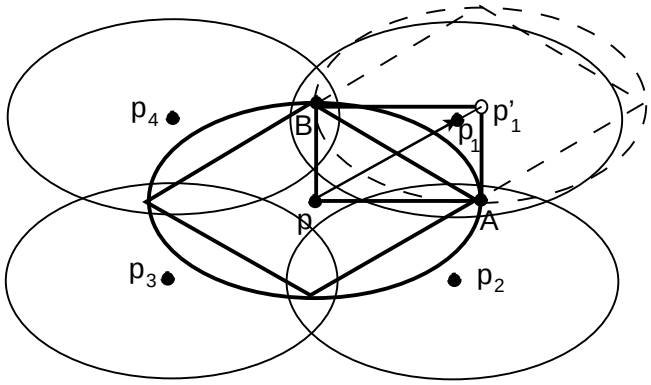



Figure 3-8: Illustration of choosing next kernel center pixel

### 3.4.2 Kernel Packing

We now determine the location of the next patch. The spacing of the patches should satisfy two conditions: i) cover all pixels in a depth layer for reconstruction, and ii) overlap to some controlled extent to avoid boundary artifacts. However, too large an overlap will increase the number of patches and computation complexity.

To compute an appropriate distance between two neighboring patches, we first assume that the to-be-computed kernels have the same shape and size as the just computed one. Further, we approximate a kernel's ellipse shape with a *diamond*, where the diamond's corners are the ellipse' endpoints along the major and minor axis. The appropriate spacing of the patches is then computed as the spacing of the diamonds when packed to fill a plane without overlap. It is known that a diamond—itself a concatenation of two identical triangles connected back-to-

back—can be used to fill a plane without gap—a process called *tessellation*[2]. Hence the spatial areas enclosed by the tiled diamonds alone are guaranteed to cover all pixels in the 2D grid. Further, because an ellipse always properly encloses the approximating diamond, the spatial areas covered by the ellipses contain overlaps. This patch selection procedure based on diamond tiling thus satisfies both required conditions: i) cover all pixels in a depth layer for optimization, and ii) introduce overlaps to eliminate boundary artifacts. See Fig. 3-8 for an illustration.

## 3.5 Graph Construction

Having identified a subset of pixels in the same depth layer suitable for interpolation, we next define a graph $\mathcal{G}$ connecting these pixels for graph-based optimization as follows. Each pixel in the kernel ellipse is represented as a node in the graph $\mathcal{G}$. We draw four edges between each pixel and its four nearest neighbors in terms of Euclidean distance, and each edge weight $e_{p,q}$ between two pixels $p$ and $q$ is computed as:

$$\begin{aligned} e_{p,q} &= w_{p,q} v_{p,q}, \\ w_{p,q} &= \exp\left\{ -\frac{\left\| I_p - I_q \right\|_2^2}{\sigma_I^2} \right\}, \\ v_{p,q} &= \exp\left\{ -\frac{\left\| \mathbf{C}_p - \mathbf{C}_q \right\|_2^2}{\sigma_{\mathbf{C}}^2} \right\} \end{aligned} \tag{3.8}$$

where $I_p$ and $C_p$ are the intensity and Cartesian grid coordinate for pixel $p$ respectively. $\sigma_I$ and $\sigma_{\mathbf{C}}$ are chosen parameters. This exponential edge weight assignment is similar to one used in bilateral filter [47].

Having constructed the graph, we compute the graph Laplacian matrix $\mathbf{L}$ described in Section 2.2.4 to define the prior probability of the underlining signal $\mathbf{s}^o$, a vector composed of the $n$ pixels in the graph $[s_1^o, \ldots, s_n^o]$:

$$Pr(\mathbf{s}^o) = C \exp\left\{ -d\ \mathbf{s}^{oT} \mathbf{L}^h \mathbf{s}^o \right\} \tag{3.9}$$

[2]https://en.wikipedia.org/wiki/Tessellation

where $d$ is a chosen parameter for the distribution and $C$ is chosen so that the distribution integrates to one. The graph-signal smoothness prior promotes signals that are smooth with respect to the defined graph.

## 3.6 MAP Formulation

We now derive the objective of our MAP formulation. Without loss of generality, let the $n_s$ synthesized pixels, $n_p$ previously interpolated pixels and $n - n_s - n_p$ empty pixels in the kernel be arranged in order in signal $\mathbf{s}$; *i.e.*, the signal has initial observation $[s_1, \ldots, s_{n_s}, p_{n_s+1}, \ldots, p_{n_s+n_p}, 0, \ldots, 0]$. Let $\mathbf{u}_i$'s be a set of $n_s + n_p$ length-$n$ unit vectors, $[0, \ldots, 0, 1, 0, \ldots, 0]$, where the single non-zero entry is at position $i$. As previously discussed, the error between the synthesized pixels in $\mathbf{s}$ and the underlining signal $\mathbf{s}^o$ follows a Laplacian distribution. Thus, our *likelihood* will be the product of a series of probability density functions:

$$Pr(\mathbf{s} \mid \mathbf{s}^o) = \prod_{i=1}^{n_s} \frac{1}{2l} \exp\left\{-\frac{\|\mathbf{u}_i^T \mathbf{s}^o - s_i\|_1}{l}\right\} \tag{3.10}$$

Given the defined prior and likelihood, we now formulate a MAP estimation problem to find the most probable $\mathbf{s}$, where the posterior probability is replaced by the product of the prior probability and the likelihood:

$$Pr(\mathbf{s}^o|\mathbf{s}) \propto Pr(\mathbf{s}|\mathbf{s}^o) \times Pr(\mathbf{s}^o) = \prod_{i=1}^{n_s} \frac{1}{2l} \exp\left\{-\frac{\|\mathbf{u}_i^T \mathbf{s}^o - s_i\|_1}{l}\right\} \cdot C \exp\left\{-p\, \mathbf{s}^{oT} \mathbf{L}^h \mathbf{s}^o\right\} \tag{3.11}$$

Then the estimation $\hat{\mathbf{s}}$ of $\mathbf{s}^o$ is computed as the argument that minimizes the negative log of our formulated objective:

$$\min_{\hat{\mathbf{s}}} \sum_{i=1}^{n_s} \|\mathbf{u}_i^T \hat{\mathbf{s}} - s_i\|_1 + \mu\, \hat{\mathbf{s}}^T \mathbf{L}^h \hat{\mathbf{s}} \tag{3.12}$$

where for simplicity $\mu$ replaces various constants in previous formulation.

Further, during a particular patch-based optimization, the optimized solution $\hat{\mathbf{s}}$ should be consistent with previously optimized pixels $p_{n_s+1}, \ldots, p_{n_s+n_p}$ in the ellipse. We apply the following constraint on the difference between $\hat{\mathbf{s}}$ and previously optimized pixels:

$$\sum_{j=n_s+1}^{n_s+n_p} \|\mathbf{u}_j^T \hat{\mathbf{s}} - p_j\|_2^2 \leq \tau \tag{3.13}$$

We describe an efficient algorithm to solve (3.12) with constraint (3.13) next.

## 3.7 Algorithm Development

### 3.7.1 Lagrangian Relaxation

To solve the formulated constrained optimization problem directly is difficult, and so we convert it into an unconstrained optimization via Lagrangian relaxation:

$$\min_{\hat{\mathbf{s}}} \sum_{i=1}^{n_s} \|\mathbf{u}_i^T \hat{\mathbf{s}} - s_i\|_1 \; + \; \mu \, \hat{\mathbf{s}}^T \mathbf{L}^h \hat{\mathbf{s}} \; + \; \nu \sum_{j=n_s+1}^{n_s+n_p} \|\mathbf{u}_j^T \hat{\mathbf{s}} - p_j\|_2^2 \tag{3.14}$$

where weight parameter $\nu > 0$ must be chosen so that the original constraint (3.13) is met.

Our new objective is then to minimize a weighted sum of: i) the $l_1$-norm of the difference between interpolated signal $\mathbf{x}$ and $n_s$ synthesized pixels $s_i$'s , ii) smoothness prior $\hat{\mathbf{s}}^T \mathbf{L}^h \hat{\mathbf{s}}$, and iii) penalty term to penalize the inconsistency between the interpolated signal and $n_p$ previously optimized pixels $p_j$'s.

### 3.7.2 Iterative reweighted Least Square Algorithm

The objective (3.14) is a combination of one $l_1$-norm term and two $l_2$-norm terms. To solve the problem efficiently, we leverage on the idea of *iterative reweighted least square* (IRLS) [44], where the $l_1$-norm fidelity term of $\hat{\mathbf{s}}$ in (3.12) is replaced by a weighted $l_2$-norm:

$$\min_{\hat{\mathbf{s}}} \sum_{i=1}^{n_s} w_i \|\mathbf{u}_i^T \hat{\mathbf{s}} - s_i\|_2^2 + \mu\, \hat{\mathbf{s}}^T \mathbf{L}^h \hat{\mathbf{s}} + \nu \sum_{j=n_s+1}^{n_s+n_p} \|\mathbf{u}_j^T \hat{\mathbf{s}} - p_j\|_2^2 \tag{3.15}$$

where the weights $w_i$'s are calculated as:

$$w_i = \frac{1}{\|\mathbf{u}_i^T \hat{\mathbf{s}} - s_i\|_1 + \epsilon'} \tag{3.16}$$

where $\epsilon' > 0$ is used for numerical stability.

With weights calculated by (3.16), the weighted $l_2$-norm (3.15) minimizing solution $\mathbf{s}^*$ will coincide with the solution $\hat{\mathbf{s}}$ of the original $l_1$-norm minimization. However, $\hat{\mathbf{s}}$ is unknown beforehand. To find appropriate weights $w_i$'s for (3.15) to approximate (3.12), we design an algorithm, where (3.15) is solved iteratively, with weights $w_i^{(t)}$'s at iteration $t$ computed as:

$$w_i^{(t)} = \frac{1}{\|\mathbf{u}_i^T \mathbf{s}^{*(t-1)} - s_i\|_1 + \epsilon'} \tag{3.17}$$

where $\mathbf{s}^{*(t-1)}$ is the solution of the previous iteration $t-1$.

The idea is that in the iterative algorithm, one can assume each iteration's $\mathbf{s}^{*(t)}$ serves as a good estimate to optimal solution $\hat{\mathbf{s}}$. Hence, we can define the weights using (3.17), so the iteratively weighted $l_2$-norm can mimic the $l_1$-norm. Algorithm 1 shows how we adopt IRLS for optimizing (3.15). Each iteration of the algorithm is an unconstrained quadratic programming problem, which can be solved efficiently in closed form:

$$\mathbf{s}^{*(t)} = (\mathbf{W}^{(t)} + \mathbf{V} + \mu \mathbf{L}^h)^{-1} (\mathbf{W}^{(t)} + \mathbf{V})\mathbf{s} \tag{3.18}$$

where $\mathbf{W}^{(t)}$ is a diagonal matrix with $w_i^{(t)}$'s as its first $n_s$ diagonal elements, and $\mathbf{V}$ is another diagonal matrix whose $(n_s+1)$-th through $(n_s+n_p)$-th diagonal elements are $\nu$. The initial weights are calculated with $\mathbf{s}_0$ where the rendered pixels are kept and the missing pixels are filled with bilinear interpolation.

**Algorithm 1** Iterative algorithm to solve weighted $l_2$-norm minimization

$\mathbf{s}' \leftarrow \mathbf{s}_0$;
**while** true **do**
  $w_i \leftarrow 1/(\|\mathbf{u}_i^T\mathbf{s}' - s_i\|_1 + \epsilon')$;
  $\mathbf{s}^* = (\mathbf{W} + \mathbf{V} + \mu\mathbf{L}^h)^{-1}(\mathbf{W} + \mathbf{V})\mathbf{s}$
  **if** round($\mathbf{s}'$) equals round($\mathbf{s}^*$) **then**
    return round($\mathbf{s}^*$)
  **else**
    $\mathbf{s}' \leftarrow \mathbf{s}^*$
  **end if**
**end while**

#### Alternative Objective

As the IRLS can require many iterations before it converges, the $l_1$-norm formulation can be further reduced to an unweighted $l_2$-norm optimization problem shown below for computation reason:

$$\min_{\hat{\mathbf{s}}} \sum_{i=1}^{n_s} \|\mathbf{u}_i^T\hat{\mathbf{s}} - s_i\|_2^2 + \mu\,\hat{\mathbf{s}}^T\mathbf{L}^h\hat{\mathbf{s}} + \nu \sum_{j=n_s}^{n_s+n_p} \|\mathbf{u}_j^T\hat{\mathbf{s}} - t_j\|_2^2 \tag{3.19}$$

The intuition is that the $l_2$ norm can also serves as a fidelity term to penalize $\hat{\mathbf{s}}$ that deviates from $\mathbf{s}$ for $i = 1 \ldots n_s$. Further, if the reference texture and depth images are coarsely quantized during compression, the noise of the synthesized image is dominated by quantization, and modeling noise as the $l_1$-norm is no longer necessary.

In this alternative formulation, (4.11) is again an unconstrained quadratic programming problem with a closed form solution, and thus, can be solved efficiently. In the experiment, the unweighted $l_2$-norm formulation is referred as UL2A, and applied when large QP is used for efficient computation. We will discuss it in detail in Section 3.8.

### 3.7.3 Selection of Smoothing Parameters

The amount of smoothness applied during the optimization can be adapted locally via adjustments to parameters $\mu$ and $h$ in (3.15) or (4.11). First, $h$, the power of

the graph Laplacian, $\mathbf{L}$, means a signal should be smooth with respect to its $h$-hop neighbors. In this work, we select $h$ to be proportional to the major radius $a$ of the adaptive kernel ellipse. The rationale is that an elongated ellipse means more pixels geometrically farther from the target pixel is included in the kernel, and our $h$ selection allows the filtering to smooth over more pixels when a strong patch gradient is detected (large $\lambda_2$), resulting in a sharper textural edge. In particular, we set $h$ to be the Manhattan distance between the target pixel and the pixel at the end of the long axis for the experiment.

Unlike our previous work [3], where parameter $\mu$—weighting the importance of the smoothness prior relative to the fidelity term—is chosen globally and heuristically, our new proposal assigns different $\mu$'s for different patches. For each patch $\mathcal{P}_v$ around a target pixel $p_v$ in the virtual view, we first find the corresponding center pixel $p_r$ in the reference view by reverse DIBR, and then draw a reduced-size version of $\mathcal{P}_v$ around $p_r$ to define $\mathcal{P}_r$. Then, we remove a selection of pixels in $\mathcal{P}_r$ and test our interpolation method using a set of parameter candidates $\mu$'s. The $\mu$ value that leads to best interpolation quality is then used to interpolate $\mathcal{P}_v$.

**Complexity Analysis**

We now discuss the complexity of our patch-based approach as compared to the pixel-based approach LARK [24]. First, our adaptive kernel is based on eigen-decomposition of a $2 \times 2$ structure tensor, while LARK performed singular value decomposition (SVD) of a much larger matrix. Both eigen-decomposition and SVD require $O(n^3)$ computation time, where $n$ is the larger dimension of the target matrix. Thus for each computed kernel our implementation is significantly faster.

Second, LARK is *pixel*-based, which means that a kernel is constructed for *every* pixel for reconstruction. In contrast, our optimization is *patch*-based, and all the pixels in a patch are reconstructed simultaneously. Thus the number of calculated adaptive kernels in our method is reduced compared to LARK by a factor equals to the average non-overlapping area inside a diamond, which is approximately 6 in our experiments. We can thus estimate that our patch-based approach is no more

than 1/6 the computation cost of LARK. (Note that our previous method [3] is also pixel-based, and thus we expect roughly the same speedup factor by employing a patch-based optimization proposed here.)

Finally, we note that our scheme is inherently local, which is much faster than non-local schemes like nonlocal means (NLM) [49] that rely on searches of similar patches in distant spatial areas and thus is computationally much more complex.

## 3.8 Experiment and Results

### 3.8.1 Experimental Setup

To demonstrate the performance of our proposed $z$-dimensional image synthesis method, we conducted extensive experiments using the Middlebury's 2003, 2005 and 2014 datasets[3] and Nagoya datasets[4], which are multiview image sequences captured indoor using an array of cameras shifted along the $x$-dimension. We consider a reduction of the distance between the observer and the nearest object by half, which means the spatial resolution of the nearest object can be increased by 2x.

Without an actual array of cameras set up along both $x$- and $z$-dimension to capture different viewpoint images, we performed the following to establish ground truth. Assuming a captured image $v_0$ is the *near-camera* image, we first synthesized an image $v_r$ as observed from the *far-camera*—located at roughly twice the distance from the 3D scene as the near-camera—via DIBR using $v_0$ as reference. This near-to-far DIBR view synthesis typically generates no expansion holes (pixel sampling in the reference view is sufficient), but has large out-of-view holes (and some disocclusion holes), since the near-camera reference views have narrower fields-of-view. To avoid the problem of filling out-of-view holes, we only considered the available field-of-view in a synthesized far-camera viewpoint image as the spatial area of interest. Examples of synthesized far-camera texture and depth images are shown

[3]http://vision.middlebury.edu/stereo/data
[4]http://http://www.fujii.nuee.nagoya-u.ac.jp/multiview-data

in Fig. 3-9.

(a) `Art` color map (b) `Art` depth map

Figure 3-9: Example of DIBR-synthesized far-camera color and depth images

Using these synthesized narrower field-of-view far-camera images as new references, we first compressed the color and depth maps using H.264 [50] with different quantization parameters (QP). We then synthesized back the near-camera images $\hat{v}_0$'s via our proposed $z$-dimensional pixel mapping, identified and filled expansion holes using our proposed scheme, and completed disocclusion holes using an existing disocclusion hole filling algorithm [5]. We compared our constructions to $v_0$ to compute PSNR and two other quality metrics SSIM [35] and 3DSwIM [45] to evaluate the quality of our constructed DIBR-synthesized images.

For comparison, we employed seven competing schemes. In the first scheme called `VSRS+`, we modified VSRS software version 3.5 [5] to use a single reference view for pixel mapping, and then called the default inpainting scheme in VSRS to fill in all missing pixels. The remaining three schemes first employed our proposed $z$-dimensional DIBR for initial pixel mapping, identified expansion holes using our proposed depth layering method, then completed expansion holes using different interpolation methods. *Note that without our proposed depth layering strategy to properly identify expansion holes, these three schemes would suffer from foreground / background confusion, as observed in the rendering results of `VSRS+`.* `Bilinear` is a

[5]While VSRS has been updated to version 4.0, the view synthesis component remains the same as version 3.5.

conventional bilinear interpolation scheme [51]. As discussed in Section 1.1, if a triangular mesh is encoded at the sender instead of the color-plus-depth format [10], then at the receiver the on-grid pixels will be linearly interpolated using nearest off-grid triangular end points. Thus `Bilinear` is representative of the rendering quality for this coding format. `Cubic+TGV` first interpolates the expansion holes via bicubic interpolation [51], then enhances the result via *Total Generalized Variation* (TGV) [52]. `LARK` is the kernel-regression based technique in [24] and can be considered the state-of-the-art among local interpolation methods. While there are recent image interpolation methods based on patch clustering, non-local methods [53, 54] or dictionary learning [55], their complexity are significantly higher than local methods. Hence we do not compare against them here for complexity reason.

Further, we compared with three methods in the literature that can be used for expansion hole filling during large $z$-dimensional camera movements, including two pixel-enlargement methods proposed in the EU project DIOMEDES [56] and MUSCADE [57], and the back-projection based method proposed in [58]. We denote these three techniques by `DIO`, `MUS` and `BP` respectively in the sequel.

In Section 3.7 we described two algorithms, `IRLS` in (3.15) and `UL2A` in (4.11), where the former employs a more accurate statistical noise model and the latter is computationally faster. We tested the two algorithms using color / depth image pairs compressed with various QPs. Table 3.1 shows the difference in PSNR between the two algorithms, computed between images constructed using our algorithms and the ground truth images for all non-disocclusion pixels (DIBR-mapped pixels from reference view and interpolated expansion hole pixels). Table 3.1 shows that `IRLS` is up to 0.22dB better than `UL2A` when $QP = 4$. As QP increases, the PSNR gain of IRLS over UL2A diminishes. The reason is that the quantization noise in compressed color and depth maps becomes dominant compared to rounding noise when QP becomes large. On the other hand, `IRLS` takes about three iterations to converge on average, *i.e.*, it requires three times the computation cost compared to `UL2A`. Thus, we used `IRLS` when QP is small ($QP \leq 12$) for better image quality and `UL2A` when QP is large ($QP > 12$) to reduce computation cost. Our proposed

Table 3.1: PSNR gain of IRLS over UL2A for different QPs

| QP | 4 | 8 | 12 | 16 | 20 | 28 | 36 |
|---|---|---|---|---|---|---|---|
| Teddy | 0.27 | 0.24 | 0.17 | 0.10 | 0.09 | 0.01 | 0.01 |
| Laundry | 0.22 | 0.16 | 0.11 | 0.08 | 0.05 | 0.02 | 0.01 |
| Art | 0.28 | 0.19 | 0.14 | 0.09 | 0.05 | 0.01 | 0.01 |
| Dolls | 0.19 | 0.15 | 0.13 | 0.08 | 0.04 | 0.02 | 0.00 |
| Moebius | 0.17 | 0.13 | 0.11 | 0.07 | 0.03 | -0.01 | -0.01 |
| Reindeer | 0.20 | 0.14 | 0.12 | 0.08 | 0.04 | 0.02 | 0.01 |
| Motorcycle | 0.21 | 0.18 | 0.14 | 0.06 | 0.04 | 0.04 | 0.02 |
| Vintage | 0.19 | 0.20 | 0.16 | 0.07 | 0.03 | 0.02 | 0.01 |

scheme is called AGFT+ in the sequel.

### 3.8.2 Experimental Comparison

#### Numerical Comparison for DIBR-synthesized Pixels

We first compare quality of DIBR-synthesized pixels with and without our proposed optimization, denoted as DIBR and AGFT+, which are shown in Fig. 3-10. Numerical PSNR values when $QP = 4$ and $QP = 16$ are listed in Table 4.1. We observe that, using our proposed scheme, PSNR can be improved by up to 0.25dB when $QP = 4$, showing that the pixels synthesized by DIBR and degraded by rounding noise can by effectively restored by our algorithm. The gain diminishes as QP increases, however, as quantization becomes coarser and details become harder to recover.

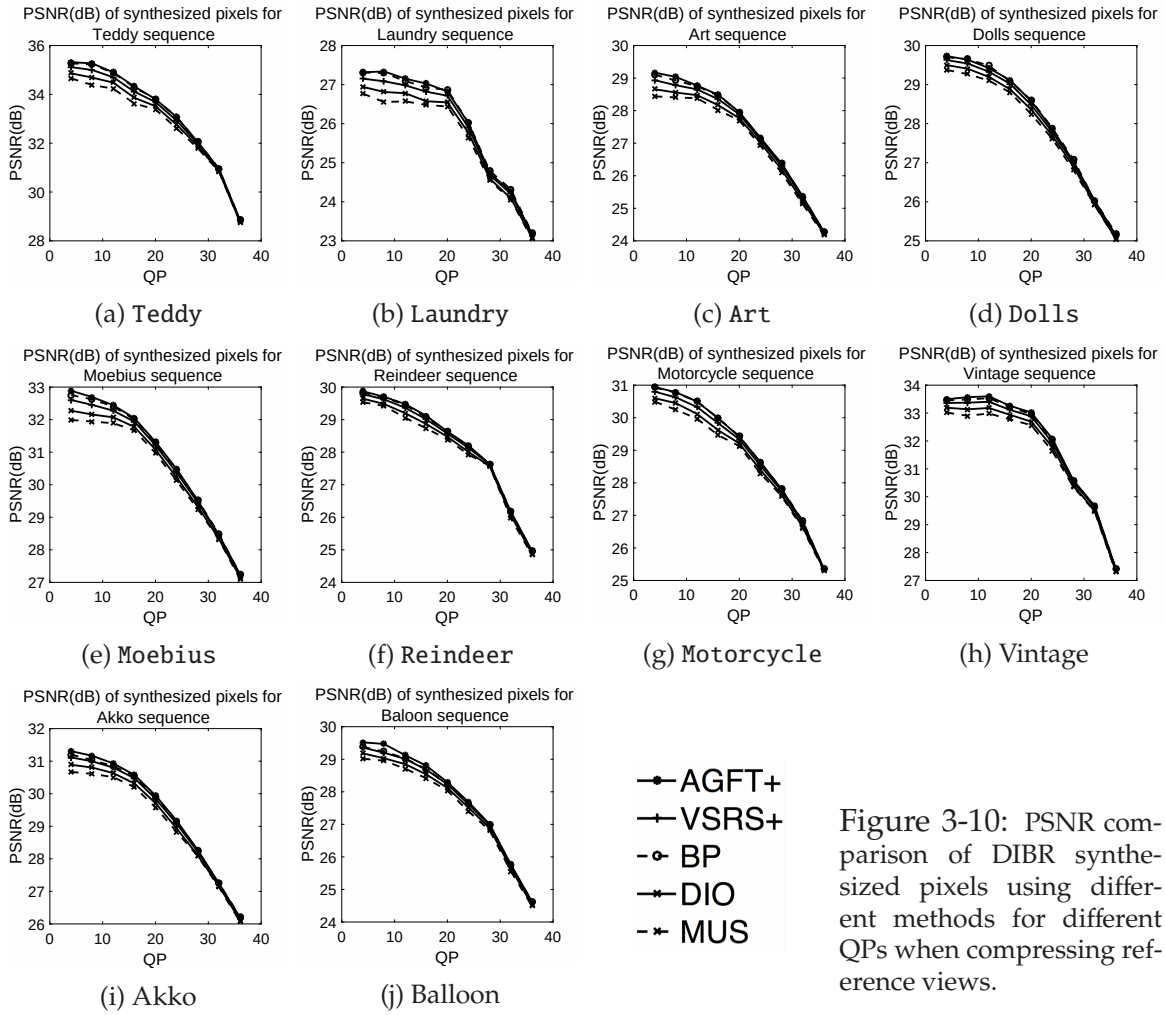
We observe that the pixel-enlargement methods DIO and MUS have even lower PSNR value than DIBR as they inevitably introduces blurring in the synthesized area, degrading the image quality.

#### Numerical Comparison for Expansion Holes

We next exam reconstruction quality of expansion holes: denoised DIBR-synthesized pixels and interpolated expansion hole pixels. As shown in Fig. 3-11 and Table 3.3, 3.4, for all eight sequences, Bilinear, Cubic+TGV, LARK, AGFT+ outperformed VSRS+ dramatically by up to 2.75dB, 2.90dB, 4.22dB and 4.01dB respectively. This demonstrates that the correct identification of expansion holes and subsequent interpola-

Table 3.2: PSNR comparison of DIBR synthesized pixels using different methods given compressed reference views

| | QP=4 | | | | | QP=16 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | DIBR | AGFT+ | BP | DIO | MUS | DIBR | AGFT+ | BP | DIO | MUS |
| Teddy | 35.10 | **35.34** | 35.27 | 34.86 | 34.66 | 34.10 | **34.30** | 34.28 | 33.86 | 33.51 |
| Laundry | 27.15 | **27.30** | **27.30** | 26.94 | 26.77 | 26.80 | **27.01** | 26.93 | 26.57 | 26.54 |
| Art | 28.85 | 29.08 | **29.11** | 28.66 | 28.44 | 28.23 | 28.37 | **28.46** | 28.17 | 27.77 |
| Dolls | 29.59 | 29.64 | **29.70** | 29.51 | 29.37 | 28.93 | 29.05 | **29.08** | 28.88 | 28.36 |
| Moebius | 32.61 | **32.86** | 32.75 | 32.27 | 31.99 | 31.93 | **32.02** | 32.00 | 31.78 | 31.08 |
| Reindeer | 29.77 | **29.83** | **29.83** | 29.64 | 29.54 | 29.00 | **29.12** | 29.07 | 28.86 | 28.47 |
| Motorcycle | 30.80 | **30.93** | 30.92 | 30.60 | 30.49 | 29.82 | **30.00** | 29.98 | 29.62 | 29.21 |
| Vintage | 33.37 | **33.49** | 33.45 | 33.19 | 33.02 | 33.11 | **33.25** | 33.21 | 32.94 | 32.69 |
| Akko | 31.12 | **31.31** | 31.20 | 30.89 | 30.66 | 30.48 | **30.58** | 30.55 | 30.32 | 29.71 |
| Balloon | 29.36 | **29.52** | 29.28 | 29.18 | 29.02 | 28.68 | **28.80** | 28.69 | 28.54 | 28.12 |




(a) Teddy (b) Laundry (c) Art (d) Dolls

(e) Moebius (f) Reindeer (g) Motorcycle (h) Vintage

(i) Akko (j) Balloon

Figure 3-10: PSNR comparison of DIBR synthesized pixels using different methods for different QPs when compressing reference views.

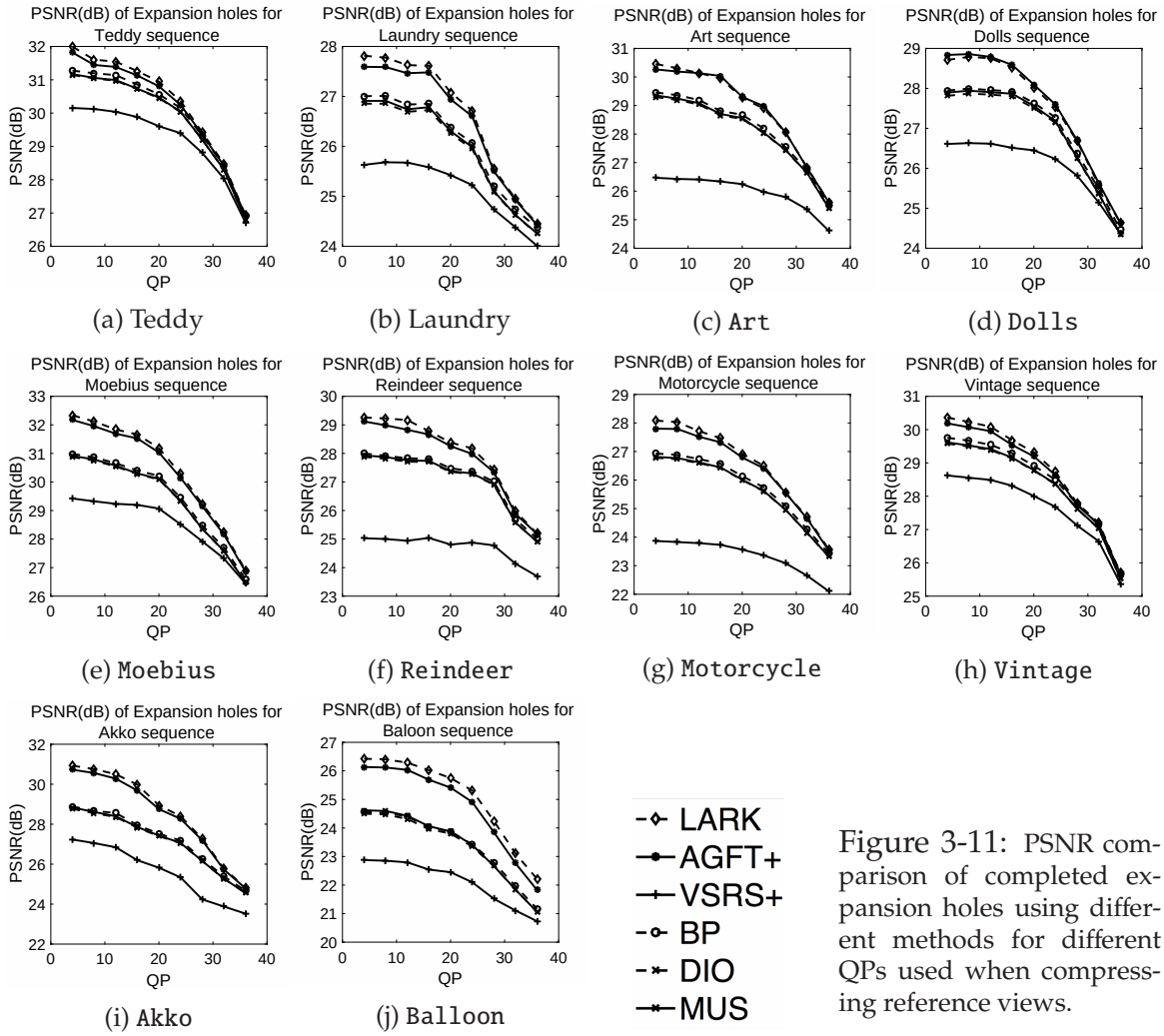
tion are crucial for $z$-dimensional DIBR image synthesis. Also, we see that AGFT+ outperformed Bilinear and Cubic+TGV by up to 1.84dB and 1.54dB, showing that by using our proposed interpolation method, we can achieve better image quality than the common image interpolation techniques. Further, our method AGFT+ has comparable PSNR numbers as LARK while doing so at a much reduced complexity. We will show later that AGFT+ is actually better than LARK in the other two quality metrics. Comparing with BP, MUS and DIO, we observe that AGFT+ outperforms them by up to 1.89dB, 1.95dB and 2.01dB respectively in the expansion hole area due to our more advanced interpolation technique. Finally, comparing to our previous method AGFT [3] shown in gray, we see that the large reduction in computation complexity results in a very small reduction in synthesis quality.

Table 3.3: PSNR comparison for completed expansion holes using different methods, given compressed reference views at QP = 4

| | $z$-dimensional DIBR, Depth Layering | | | | | BP | MUS | DIO | VSRS+ |
|---|---|---|---|---|---|---|---|---|---|
| | AGFT | AGFT+ | LARK | Cubic+TGV | Bilinear | | | | |
| Teddy | 31.86 | 31.82 | **31.99** | 31.16 | 31.02 | 31.28 | 31.17 | 31.14 | 30.14 |
| Laundry | 27.62 | 27.58 | **27.80** | 26.91 | 26.74 | 27.00 | 26.91 | 26.85 | 25.63 |
| Art | 30.34 | **30.44** | 30.26 | 29.33 | 29.15 | 29.45 | 29.35 | 29.30 | 26.43 |
| Dolls | 28.80 | **28.83** | 28.70 | 27.83 | 27.59 | 27.95 | 27.90 | 27.81 | 26.60 |
| Moebius | 32.16 | 32.16 | **32.34** | 30.87 | 30.69 | 30.99 | 30.91 | 30.87 | 29.41 |
| Reindeer | 29.08 | 29.11 | **29.25** | 27.89 | 27.68 | 30.00 | 27.93 | 27.88 | 25.01 |
| Motorcycle | 27.79 | 27.78 | **28.08** | 26.82 | 26.71 | 26.93 | 26.81 | 26.78 | 23.86 |
| Vintage | 30.20 | 30.19 | **30.36** | 29.63 | 29.53 | 29.76 | 29.62 | 29.58 | 28.62 |
| Akko | 30.76 | 30.73 | **30.92** | 28.77 | 28.54 | 28.89 | 28.84 | 28.77 | 27.23 |
| Balloon | 26.13 | 26.14 | **26.43** | 23.95 | 23.57 | 24.56 | 24.62 | 24.50 | 22.87 |

Table 3.4: PSNR comparison for completed expansion holes using different methods, given compressed reference views at QP = 16

| | $z$-dimensional DIBR, Depth Layering | | | | | BP | MUS | DIO | VSRS+ |
|---|---|---|---|---|---|---|---|---|---|
| | AGFT | AGFT+ | LARK | Cubic+TGV | Bilinear | | | | |
| Teddy | 31.16 | 31.11 | **31.26** | 30.76 | 30.68 | 30.86 | 30.71 | 30.73 | 29.87 |
| Laundry | 27.48 | 27.46 | **27.60** | 26.80 | 26.62 | 26.85 | 26.73 | 26.79 | 25.58 |
| Art | 29.95 | 29.94 | **30.00** | 28.93 | 28.78 | 28.80 | 28.66 | 28.70 | 26.32 |
| Dolls | 28.57 | **28.57** | 28.51 | 27.85 | 27.68 | 27.92 | 27.80 | 27.86 | 26.50 |
| Moebius | 31.49 | 31.50 | **31.65** | 30.31 | 30.18 | 30.40 | 30.27 | 30.30 | 29.18 |
| Reindeer | 28.62 | 28.64 | **28.77** | 27.74 | 27.59 | 27.81 | 27.69 | 27.74 | 25.04 |
| Motorcycle | 27.33 | 27.31 | **27.46** | 26.48 | 26.41 | 26.57 | 26.42 | 26.44 | 23.72 |
| Vintage | 29.53 | 29.53 | **29.67** | 29.21 | 29.15 | 29.29 | 29.13 | 29.16 | 28.30 |
| Akko | 29.69 | 29.68 | **29.98** | 27.88 | 27.74 | 27.96 | 27.82 | 27.86 | 26.20 |
| Balloon | 25.71 | 25.69 | **26.02** | 23.44 | 23.09 | 24.02 | 23.96 | 24.06 | 22.54 |




(a) Teddy (b) Laundry (c) Art (d) Dolls

(e) Moebius (f) Reindeer (g) Motorcycle (h) Vintage

(i) Akko (j) Balloon

Figure 3-11: PSNR comparison of completed expansion holes using different methods for different QPs used when compressing reference views.

### Overall Numerical and Visual Comparison

Instead of PSNR, in addition we use two other image quality metrics to evaluate the performance of our $z$-dimensional image synthesis: SSIM [35], the most commonly used image visual quality assessment metric, and 3DSwIM [45], a recently proposed metric dedicated to artifacts detection in 3D synthesized views. The numerical results are shown in Table 3.5 and 3.6.

Next, we examine the constructed image quality visually. In Fig. 3-12, we show the constructed images and closed-up patches by different methods. First, we see visually in Fig. 3-12(a) that $z$-dimensional pixel mapping caused the erroneous mixing of foreground / background pixels during DIBR. Applying inpainting algorithm naïvely to fill in all missing pixels subsequently do not lead to acceptable

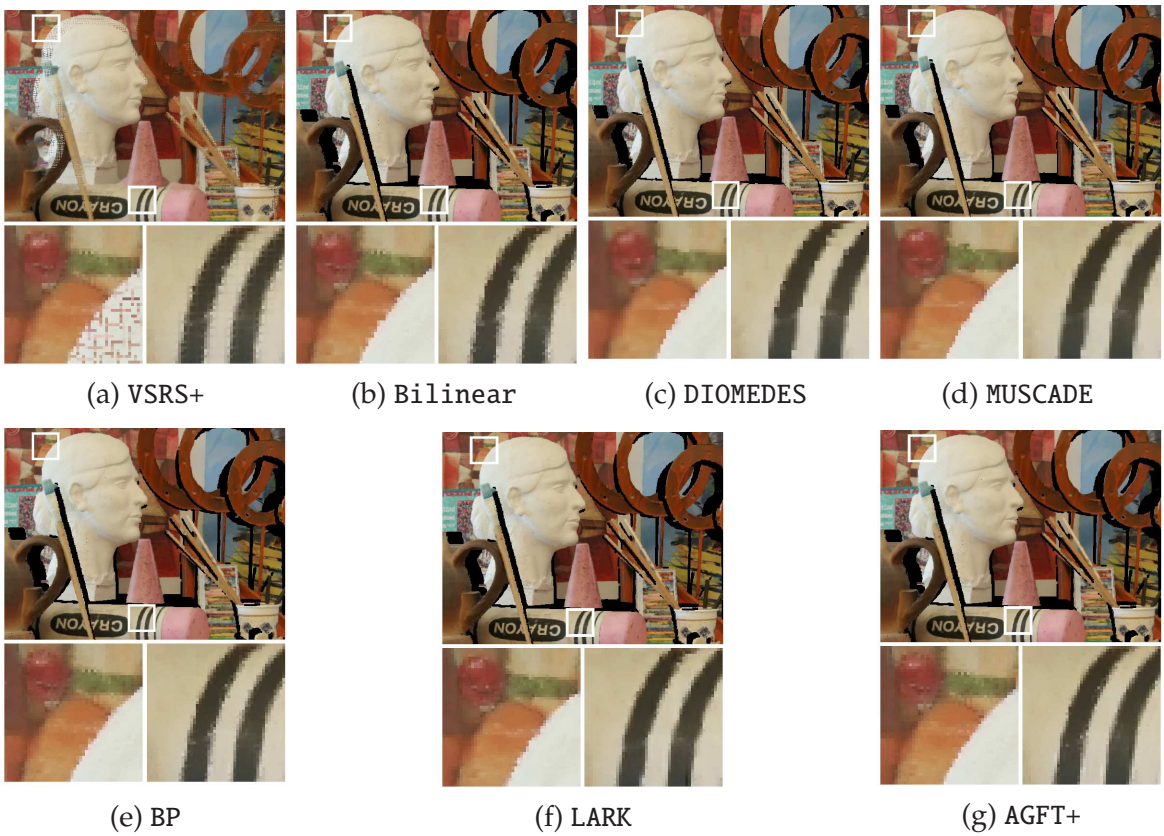
Table 3.5: SSIM comparison for synthesized images, reference view compression QP = 4

| | *z*-dimensional DIBR, Depth Layering | | | | | BP | MUS | DIO | VSRS+ |
|---|---|---|---|---|---|---|---|---|---|
| | AGFT | AGFT+ | LARK | Cubic+TGV | Bilinear | | | | |
| Teddy | 0.9206 | 0.9202 | 0.9206 | 0.9212 | 0.9202 | **0.9216** | 0.9193 | 0.9186 | 0.9077 |
| Laundry | 0.9393 | **0.9395** | 0.9342 | 0.9346 | 0.9312 | 0.9351 | 0.9323 | 0.9314 | 0.8971 |
| Art | 0.9426 | **0.9423** | 0.9386 | 0.9367 | 0.9369 | 0.9374 | 0.9338 | 0.9259 | 0.8885 |
| Dolls | 0.9231 | **0.9231** | 0.9180 | 0.9174 | 0.9163 | 0.9192 | 0.9183 | 0.9080 | 0.9031 |
| Moebius | 0.9239 | 0.9236 | **0.9242** | 0.9210 | 0.9184 | 0.9167 | 0.9085 | 0.9054 | 0.8912 |
| Reindeer | 0.9024 | **0.9026** | 0.9012 | 0.9002 | 0.8999 | 0.9021 | 0.8967 | 0.8928 | 0.8775 |
| Motorcycle | 0.9343 | **0.9341** | 0.9279 | 0.9281 | 0.9279 | 0.9269 | 0.9257 | 0.9244 | 0.8680 |
| Vintage | 0.9490 | **0.9490** | 0.9472 | 0.9460 | 0.9458 | 0.9446 | 0.9430 | 0.9409 | 0.9366 |
| Akko | 0.9494 | **0.9492** | 0.9460 | 0.9453 | 0.9479 | 0.9387 | 0.9384 | 0.9367 | 0.8932 |
| Balloon | 0.9076 | **0.9079** | 0.9061 | 0.9048 | 0.9023 | 0.9045 | 0.9012 | 0.9006 | 0.8727 |

Table 3.6: 3DSwIM comparison for synthesized images, reference view compression QP = 4

| | $z$-dimensional DIBR, Depth Layering | | | | | BP | MUS | DIO | VSRS+ |
|---|---|---|---|---|---|---|---|---|---|
| | AGFT | AGFT+ | LARK | Cubic+TGV | Bilinear | | | | |
| Teddy | 0.8459 | **0.8462** | 0.8453 | 0.8392 | 0.8384 | 0.8461 | 0.8398 | 0.8339 | 0.8425 |
| Laundry | 0.9137 | **0.9132** | 0.8992 | 0.9060 | 0.9062 | 0.9136 | 0.9123 | 0.9042 | 0.9101 |
| Art | 0.9741 | **0.9744** | 0.9621 | 0.9684 | 0.9676 | 0.9691 | 0.9672 | 0.9651 | 0.9573 |
| Dolls | 0.9500 | 0.9499 | 0.9479 | 0.9482 | 0.9499 | **0.9512** | 0.9476 | 0.9468 | 0.9459 |
| Moebius | 0.8938 | **0.8936** | 0.8925 | 0.8873 | 0.8732 | 0.8839 | 0.8724 | 0.8715 | 0.8514 |
| Reindeer | 0.9512 | 0.9506 | 0.9524 | 0.9545 | **0.9590** | 0.9524 | 0.9533 | 0.9497 | 0.9445 |
| Motorcycle | 0.9763 | **0.9754** | 0.9698 | 0.9679 | 0.9678 | 0.9694 | 0.9672 | 0.9613 | 0.9370 |
| Vintage | 0.7595 | **0.7597** | 0.7490 | 0.7496 | 0.7494 | 0.7312 | 0.7325 | 0.7322 | 0.7461 |
| Akko | 0.8623 | **0.8619** | 0.8614 | 0.8609 | 0.8602 | 0.8617 | 0.8615 | 0.8606 | 0.8329 |
| Balloon | 0.7453 | **0.7453** | 0.7412 | 0.7386 | 0.7367 | 0.7495 | 0.7351 | 0.7312 | 0.7376 |

quality in the expansion hole areas. Second, we see in Fig. 3-12(b) and Fig. 3-12(c) that even only using the pixels in the same depth layer for interpolation, `Bilinear` and `Cubic+TGV` will introduce significant interpolation artifacts, especially on the texture edges. Finally, we see in Fig.3-12(d) that by using our proposed `AGFT+`, which is comparable to result produced by `LARK` in Fig.3-12(e), expansion holes can be filled in a visually pleasing manner. Similar results for the `doll` sequence are shown in Fig. 3-13, Fig. 3-14 and Fig. 3-15.



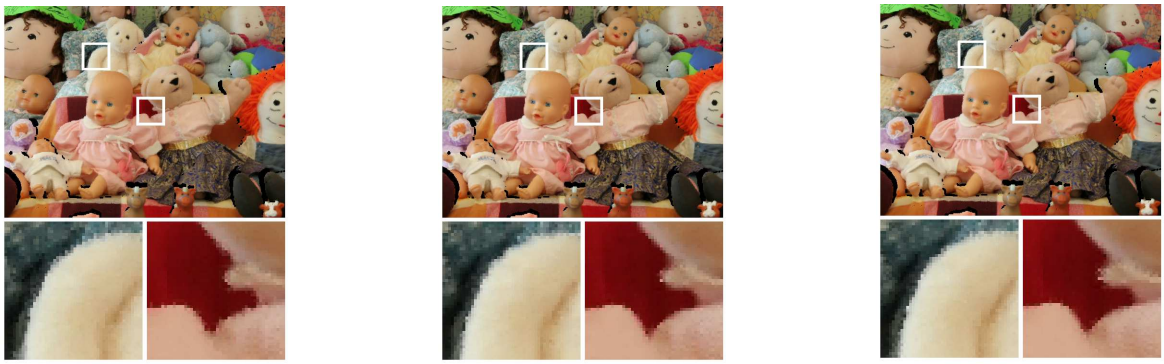
(a) `VSRS+` (b) `Bilinear` (c) `DIOMEDES` (d) `MUSCADE`

(e) `BP` (f) `LARK` (g) `AGFT+`

Figure 3-12: Visual evaluation of synthesized images for `Art` with QP = 4

Finally, we filled in disocclusion holes using an existing scheme [5] to get the complete $z$-dimensional DIBR-synthesized images. As we can see in Fig. 3-16 and Fig. 3-17, visually pleasing images can be successfully synthesized by our proposal combined with an appropriate disocclusion hole inpainting algorithm like [5].

(a) VSRS+ (b) Bilinear (c) DIOMEDES (d) MUSCADE



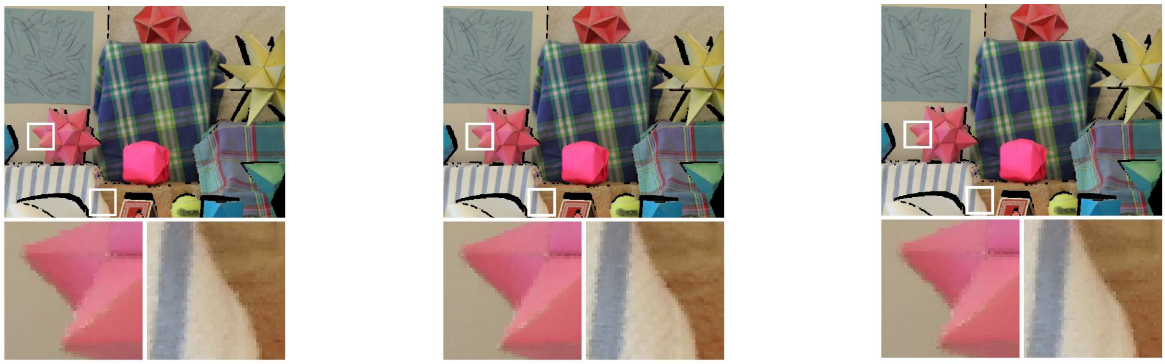
(e) BP (f) LARK (g) AGFT+

Figure 3-13: Visual evaluation of synthesized images for Dolls with QP =4

## 3.9 Chapter Summary

Unlike typical free viewpoint system that considers only synthesis of novel images of virtual views shifted horizontally along the $x$-dimension via depth-image-based rendering (DIBR), in this chapter, we consider in addition construction of $z$-dimensional DIBR-synthesized images. In such far-to-near viewpoint synthesis, there exists a new type of missing pixels called expansion holes—where objects close to the camera will increase in size and simple pixel-to-pixel mapping in DIBR from reference to virtual view will result in missing pixel areas—that demand a new interpolation scheme. In this chapter, we propose to first identify expansion holes via a depth layering procedure, then formulate a maximum a posteriori (MAP) problem to estimate the missing pixels using a graph-signal smoothness prior. We propose an iterative reweighted least square (IRLS) algorithm to solve the posed MAP problem efficiently. Experimental results show up to 4.01dB gain in PSNR

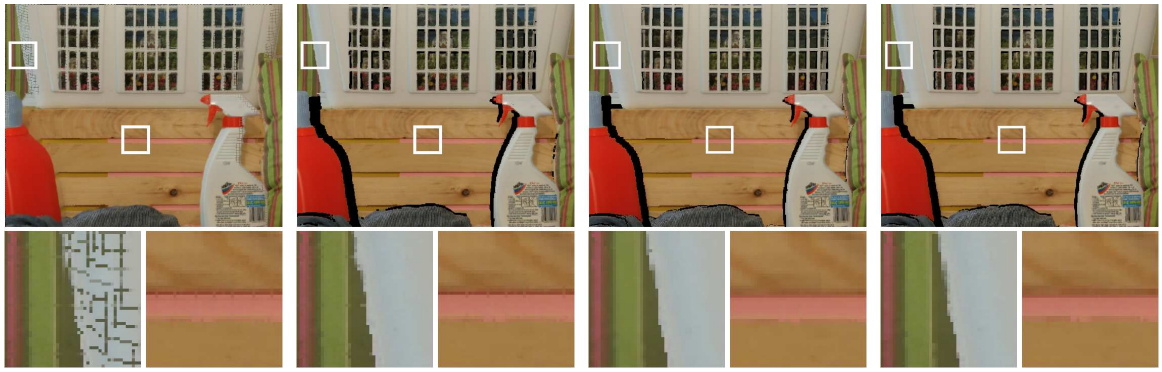
(a) VSRS+ (b) Bilinear (c) DIOMEDES (d) MUSCADE



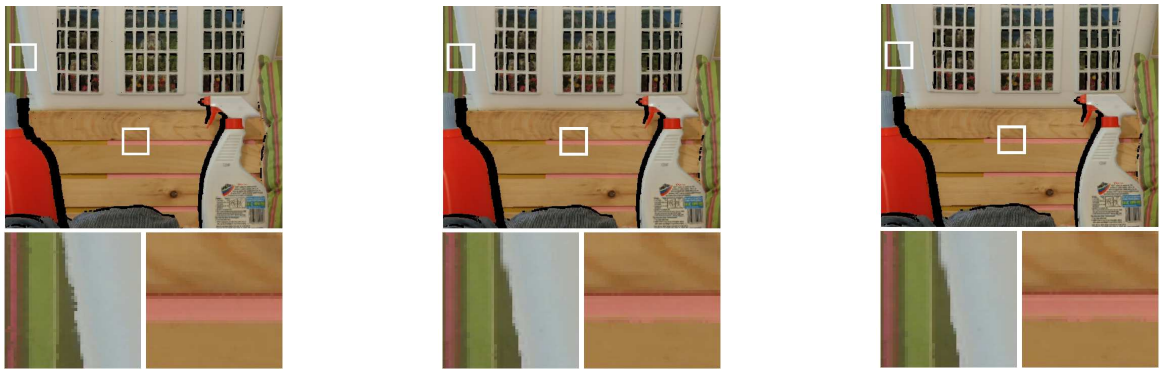
(e) BP (f) LARK (g) AGFT+

Figure 3-14: Visual evaluation of synthesized images for Moebius with QP = 4

over inpainting method employed in VSRS 3.5. While we focus on static multiview image rendering, our work can be extended to multiview video rendering, where additional requirements such as temporal consistency [7, 59] need to be considered also.



(a) VSRS+ (b) Bilinear (c) DIOMEDES (d) MUSCADE



(e) BP (f) LARK (g) AGFT+

Figure 3-15: Visual evaluation of synthesized images for Laundry with QP = 4

(b) VSRS+ (a) Proposed

Figure 3-16: Final output images for $z$-movement DIBR for art

(b) VSRS+ (a) Proposed

Figure 3-17: Final output images for $z$-movement DIBR for `dolls`

# Chapter 4

# Joint Depth Image Denoising and Interpolation via Graph Signal Processing

## 4.1 Introduction

Recent advanced depth sensing technologies have found applications in a wide range of practical scenarios, including image refocusing, virtual view synthesis via depth-image-based rendering (DIBR), and non-intrusive sleep monitoring. One notable usage is depth image sensing for unmanned aerial vehicles (UAV) like drones, where an obtained depth image is analyzed quickly to recognize obstacles in their flight paths for collision avoidance. Unlike color images, depth images acquired from actively sensed infrared samples means they are unaffected by changing lighting conditions, and thus can operate reliably in the dark. On the other hand, active infrared sensing using time-of-flight (ToF) principle also means more energy consumption, which is challenging for a battery-powered UAV already scarce in resource.

In response, in this chapter we tackle the technical challenge of fast restoration of *piecewise smooth* (PWS) images such as depth images given sparse, noisy pixel

samples—fewer the number of actively sensed samples, lower the energy cost. By PWS, we mean an image with mostly smooth surfaces interrupted by discontinuities; in the case of depth images, discontinuities correspond to boundaries between foreground objects and background. By fast, we mean processing operations that are performed *locally* on a pixel patch (thus amenable to parallel implementation) without the expensive cost of global search like nonlocal means (NLM). Further, algorithms that require a variable number of iterations depending on characteristics of the current target pixel patch to satisfy an exit condition is also not desirable. With these stringent conditions, many state-of-the-art image restoration algorithms become unsuitable for our application needs.

Leveraging on recent advances in graph signal processing (GSP), we propose a joint denoising / interpolation scheme for PWS image restoration. We first prefilter an image to obtain initial pixel estimates at each 2D grid pixel location. We then detect strong edges via spectral clustering. We decompose an image into a quadtree representation, where each leaf represents a smooth image patch. Each patch is first coarsely approximated via 2D linear regression, then finely enhanced via a local graph-based filtering operation.

## 4.2 Graph Signal Processing via Densely Constructed Graph

## 4.3 Image Model

### 4.3.1 Image Model for Piecewise Smooth Images

We now propose an image model for piecewise-smooth image restoration problem which combines a quadtree structured piecewise-linear prediction and graph Laplacian regularizer for the prediction residual. As the regularity of piecewise-smooth images can be interpreted as smooth regions separated by strong edges, we propose to approximate a piecewise-smooth image by a set of 2-D linear functions,

(a)Compressively sensed noisy signal (b)Prefiltered image (c)Quadtree decomposition

Figure 4-1: An example of the compressively sensed signal, its prefiltered image and the output of quadtree decomposition. It is shown that while the quality of the prefiltered image is far from satisfying, the general structure of the image can be roughly estimated.

whose domains are organized in the form of a quadtree.

Especially, the quadtree has square blocks as its nodes, and both the biggest and the smallest block sizes are fixed. It is so constructed that any block containing a strong edge and larger than the smallest size is further divided into four smaller blocks. As a result, only the smallest leaf blocks in the quadtree may contain a strong edge. Further, more and smaller blocks will be allocated near the edges, leading to a finer treatment for the area, which tend to have more complex characteristics. On the hand, large flat area will be covered with large blocks, within which we have more pixels to average over, producing smoother results in the block.

In the scenario of image restoration from compressively sensed signal, we will first apply the filter proposed in [24] on the the sparse and noisy input $\mathbf{z}$ for a single iteration to get $\mathbf{z}'$ as a prefiltered image. Then Canny edge detection is applied on $\mathbf{z}'$ to derive a edge map, based on which we construct the quadtree using the method described above. The procedure is illustrated by Fig. 4-1:

### 4.3.2 Piecewise Linear Function Approximation

After the quadtree decomposition, we use simple functions to approximate the signals within the blocks. In the existing research , smooth signal within a small region has been modeled by 2D polynomials. As in our proposal, the approximation only serves as a coarse estimation of the underlying signal, we further restrict the order of the polynomial to be one, i.e, 2-D linear function $f(\mathbf{c}_i)$ whose variable

$\mathbf{c_i}$ is the pixel's coordinate:

$$f_{\mathbf{G}}(\mathbf{C}_i) = \mathbf{G} \cdot [\mathbf{C}_i, 1] \tag{4.1}$$

The Plane's parameter vector ($G$) is then determined via solving least squares problem:

$$\min_{\mathbf{G}} \sum_{i \in \mathcal{P}} (y_i - f_{\mathbf{G}}(\mathbf{C}_i))^2 \tag{4.2}$$

The prediction residual $\mathbf{R}$ is then calculated as:

$$\mathbf{R} = \{r_i = y_i - f_{\mathbf{G}}(\mathbf{C_i}) | i \in \mathcal{P}\} \tag{4.3}$$

And the residuals of estimation will be regularized by the graph Laplacian, which we will introduce in details in the next subsection.

unlike the non-edge blocks, which can be easily approximated by a single linear function, the edge blocks contains multiple pieces of smooth signals, and each piece of the signals should be approximated by its own plane independently. In practice, based on our observation that it is rare for edge blocks to contain more than two pieces of smooth signals, as the they always have the smallest size, we will first divide the edge blocks into two pieces and estimate planes for the signals individually. Especially, we first use normalized cut, the image segmentation technique proposed in , to split the block into two piecewise smooth patches $\mathcal{P}_i$'s. Obviously, the quality of the approximation cannot be guaranteed when the size $|\mathcal{P}_i|$ is too small. As a result when $|\mathcal{P}_i|$ is smaller than some certain threshold, we will skip the approximation, and directly apply the graph Laplacian to regularize the sensed signal.

## 4.4 Adaptive Kernel

We adopt a block-based approach in our proposal. For each pixel $i$ in block $\mathcal{B}$, with intensity value $y_i$, we define a $p \times p$ analysis window $\mathcal{P}_i$ centered at it. Then we fix a intensity threshold $\tau_i$ and exclude the pixels that has a deviation from the target pixel $i$ larger than $\tau_i$, which provide little useful information about the target pixel's value and connecting them to the target pixel results in a bad edge. By viewpoint of the image content, these pixels are often from a diffrent object other than the target pixel's, or from another distinct part of texture, thus can be reasonably ignored during calculating the target pixel. By the above processing, we select a set $\mathcal{N}_i$ of pixels around $i$ with similar intensity values:

$$\mathcal{N}_i = j || y_j - y_i | < \tau_i, j \in \mathcal{P}_i \tag{4.4}$$

Within $\mathcal{N}_i$ we will calculate the structure tensor $S_i$ of $i$, which is computed equation 3.5. Based on $S_i$ we draw a ellipse $s$ in the same way described in 3, and denote the set of the pixels in $s$ by $\mathcal{S}_i$. The kernel we use to draw our graph is then defined as:

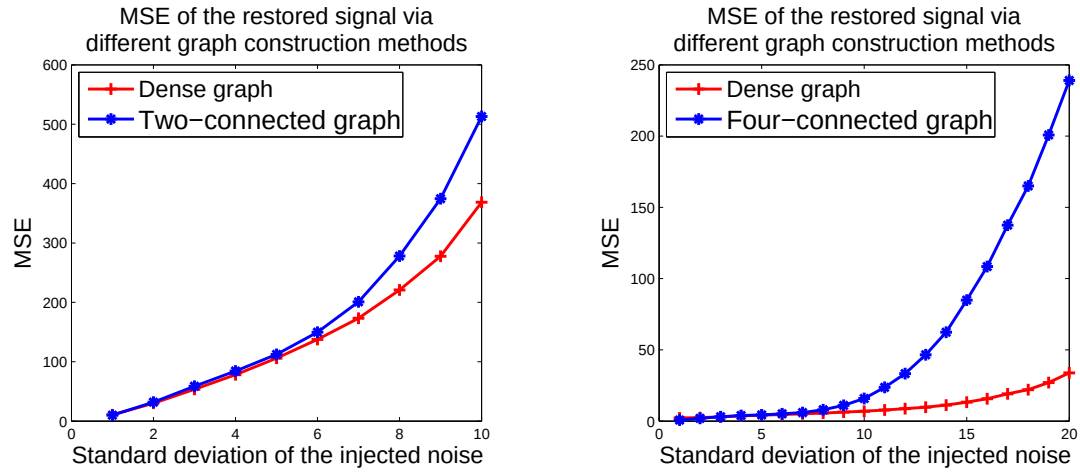
$$\mathcal{K}_i = \mathcal{S}_i \bigcap \mathcal{N}_i \tag{4.5}$$

Having defined $\mathcal{K}_i$ for each pixel $i$ in the block $\mathcal{B}$, The graph within the block $\mathcal{B}$ following:

$$\begin{gathered} \mathcal{G}_{\mathcal{B}} = \{\mathcal{B}, E\}, \\ E = \{e(i,j) \mid e(i,j) \in \mathcal{K}_i \ or \ e(i,j) \in \mathcal{K}_j, \ \ i,j \in \mathcal{B}\} \end{gathered} \tag{4.6}$$

## 4.5 Graph Construction

In this section We use the same terminologies about GSP that is defined in Chapter. 3 And the weights of the graph edges are assigned to reflect the similarities of the nodes. Especially, on image restoration, the edge weights are computed via the comparing the initial value of the pixels. While the graph Laplacian regularizer




(a) noise sensitivity test on 8x1 line signal (b) noise sensetivity test on 8x8 2D signal

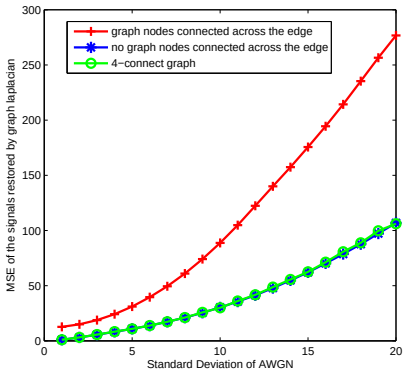
has been proved a powerful tool for image regularization by many literatures, it sufferes from the difficulty to estimate the correct graph from noisy samples. Previous works, such as [4] and [3] utilized an iterative method to approximate the correct graph via updating the edge weights by the results of the previous loop. However, the iterative methods heavily relies on the initial values. What's more the iteration cost a lot computation power.

To achieve a graph that can represent the signal faithfully in the presence of noise, and yet avoid the expensive iterations, we propose to draw more edges, i.e. create a denser graph. We had the intuition that dense graph is more robust to noises, and it won't introduce bias that seriously undermine the restoration quality. Experiments had been carried out to verify our points.

First, we show that adding more edges to the graph enhances the robustness of the graph. We did the noise sensitivity test on a 8-node 1D signal and an 8x8 2-D signal. Both signals are piecewise smooth. We fix the topology of the graph, and then use the residual regularization method proposed later to restore from the noisy sample. The results is shown in the following figures.

It is well demonstrated that the dense graph is less sensetive to the noise, which means that we can get a graph more similar to the graph constructed on the clean data.

Next, we show that adding edges properly will not introduce bias that under-




mines the restoration quality. Again we did test on two piecewise smooth signals with size 8X1 and 8x8 respectively. We add edges to the four-connected graphs, if the additive edge doesn't cross different pieces we call it a proper graph, and vice versa. The results in the following figures showed that if the additive edge is drawn properly, it will not introduce a bad bias. On the contrary, if the edge is drawn accross two distinct pieces, it will introduce a serious bias to the reconstructed signal.

### 4.5.1 Residual reconstruction with a Dense Graph Laplacian Regularizer

In this subsection, we discuss the residual reconstruction with graph Laplacian regularizer. As we discussed in Section 2.2.4, we intend to construct a graph that has additional edges to make the graph more robust to noise. And the edges should not violate the underlying structure of the image, i.e, don't connect two nodes from two distinctive pieces. And we designed the following method to construct a dense graph, which can serve our purpose well.

Having identified a subset of pixels in the same depth layer suitable for interpolation, we next define a graph $\mathcal{G}$ connecting these pixels for graph-based optimization as follows. Each pixel in the kernel ellipse is represented as a node in the graph $\mathcal{G}$. We draw edges densely between each pixel and neighbors in its adaptive kernel, and each edge weight $e_{p,q}$ between two pixels $p$ and $q$ is computed

as:

$$
\begin{aligned}
e_{p,q} &= w_{p,q} v_{p,q}, \\
w_{p,q} &= \exp\left\{ -\frac{\left\| I_p - I_q \right\|_2^2}{\sigma_I^2} \right\}, \\
v_{p,q} &= \exp\left\{ -\frac{\left\| \mathbf{C}_p - \mathbf{C}_q \right\|_2^2}{\sigma_{\mathbf{C}}^2} \right\}
\end{aligned}
\tag{4.7}
$$

where $I_p$ and $C_p$ are the intensity and Cartesian grid coordinate for pixel $p$ respectively. $\sigma_I$ and $\sigma_{\mathbf{C}}$ are chosen parameters.

This exponential edge weight assignment is similar to one used in bilateral filter [47].

Having constructed the graph, we compute the graph Laplacian matrix $\mathbf{L}$ described in Section 2.2.4 to define the prior probability of the underlining signal $\mathbf{s}^o$, a vector composed of the $n$ pixels in the graph $[s_1^o, \ldots, s_n^o]$:

$$
Pr(\mathbf{s}^o) = C \exp\left\{ -d\, \mathbf{s}^{oT} \mathbf{L}^h \mathbf{s}^o \right\} \tag{4.8}
$$

where $d$ is a chosen parameter for the distribution and $C$ is chosen so that the distribution integrates to one. The graph-signal smoothness prior promotes signals that are smooth with respect to the defined graph.

## 4.6 MAP Formulation

We now derive the objective of our MAP formulation. Without loss of generality, let the $n_s$ sensed pixels, $n - n_p$ empty pixels in the block be arranged in order in signal $\mathbf{s}$; *i.e.*, the signal has initial observation $[s_1, \ldots, s_{n_s}, 0, \ldots, 0]$. Let $\mathbf{u}_i$'s be a set of $n_s$ length-$n$ unit vectors, $[0, \ldots, 0, 1, 0, \ldots, 0]$, where the single non-zero entry is at position $i$. Assuming the error between the synthesized pixels in $\mathbf{s}$ and the underlining signal $\mathbf{s}^o$ follows a Gaussian distribution. Thus, our *likelihood* will be

the product of a series of probability density functions:

$$Pr(\mathbf{s} \mid \mathbf{s}^o) = \prod_{i=1}^{n_s} \frac{1}{2l} \exp\left\{-\frac{\|\mathbf{u}_i^T \mathbf{s}^o - s_i\|_2^2}{l}\right\} \tag{4.9}$$

Given the defined prior and likelihood, we now formulate a MAP estimation problem to find the most probable $\mathbf{s}$, where the posterior probability is replaced by the product of the prior probability and the likelihood:

$$Pr(\mathbf{s}^o|\mathbf{s}) \propto Pr(\mathbf{s}|\mathbf{s}^o) \times Pr(\mathbf{s}^o) = \prod_{i=1}^{n_s} \frac{1}{2l} \exp\left\{-\frac{\|\mathbf{u}_i^T \mathbf{s}^o - s_i\|_2^2}{l}\right\} \cdot C \exp\left\{-p\, \mathbf{s}^{oT} \mathbf{L}^h \mathbf{s}^o\right\} \tag{4.10}$$

Then the estimation $\hat{\mathbf{s}}$ of $\mathbf{s}^o$ is computed as the argument that minimizes the negative log of our formulated objective:

$$\min_{\hat{\mathbf{s}}} \sum_{i=1}^{n_s} \|\mathbf{u}_i^T \hat{\mathbf{s}} - s_i\|_1 + \mu\, \hat{\mathbf{s}}^T \mathbf{L}^h \hat{\mathbf{s}} \tag{4.11}$$

where for simplicity $\mu$ replaces various constants in previous formulation.

Our new objective is to minimize a weighted sum of: i) the $l_1$-norm of the difference between interpolated signal $\mathbf{x}$ and $n_s$ synthesized pixels $s_i$'s , ii) smoothness prior $\hat{\mathbf{s}}^T \mathbf{L}^h \hat{\mathbf{s}}$,

(4.11) is an unconstrained quadratic programming problem with the closed form solution shown below, and thus, can be solved efficiently.

$$\hat{\mathbf{s}} = (\mathbf{W} + \mu \mathbf{L}^h)^{-1} (\mathbf{W}) \mathbf{s}^o \tag{4.12}$$

where $\mathbf{W}$ is a diagonal matrix with $u_i$'s as its first $n_s$ column vectors.

Table 4.1: PSNR and SSIM comparison of restored images using different techniques

| | PSNR | | | SSIM | | |
|---|---|---|---|---|---|---|
| | Proposed | LARK | BM3D | Proposed | LARK | BM3D |
| New Tsukuba 1 | **32.94** | 32.13 | 32.53 | **0.9370** | 0.9261 | 0.9364 |
| New Tsukuba 2 | **30.50** | 29.84 | 30.34 | 0.9128 | 0.9061 | **0.9177** |
| New Tsukuba 3 | **31.96** | 31.28 | 30.82 | **0.9453** | 0.9313 | 0.9427 |
| Flowers | **31.30** | 30.04 | 30.37 | **0.9219** | 0.9113 | 0.9205 |
| Motorcycle | **27.62** | 27.00 | 27.09 | **0.8781** | 0.8670 | 0.8713 |
| Recycle | **34.01** | 33.25 | 33.70 | **0.9453** | 0.9357 | 0.9357 |

## 4.7 Experiment and Results

### 4.7.1 Depth Image Interpolation

#### Experiment Setup

We did extensive experiments to test the proposed restoration algorithm for compressive sensed depth images. The test inputs are artificially created by first degrade the public accessible depth images from Middlebury database and the "New Tsukuba" test sequence with Additive White Gaussian Noise (AWGN) and then randomly remove a major portion of the pixels in the degraded images. The proposal is then compared against BM3D and LARK. For all three methods, we used the same pre-filtered images.

#### Numerical Comparison

We add AWGN with its standard deviation equals 10 to the image and then remove 75% of pixels to create the input. Then we use our proposed scheme and the competing methods to recover the signal. As for LARK, we pick its output at the sixth iteration, which have the highest quality. For BM3D, it used the same prefiltered image used in our method as input. The numerical comparison is shown in the following table.

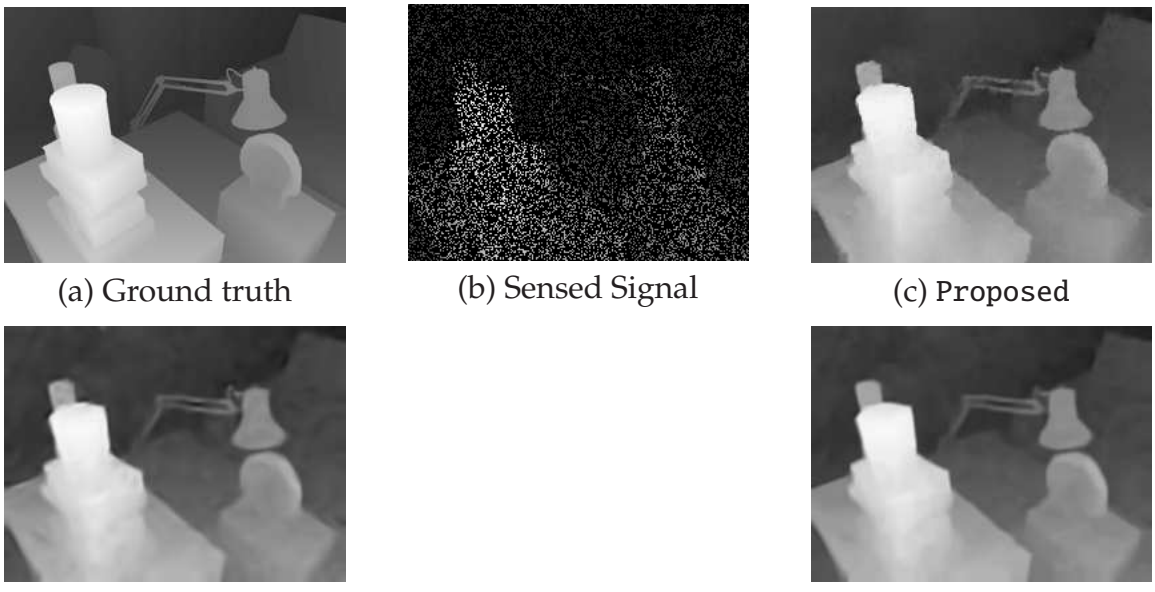
We can see that our scheme outperformed the other two methods for all six test images by PSNR. Especially, we beat LARK and BM3D by up to 1.26dB and 0.93dB respectively on Flowers. For the evaluation by SSIM, we again outperformed LARK

for all six test images, and better than BM3D on five test images except for New Tsukuba 2.

We would like to highlight that, comparing with the other two State-of-Art Algorithms, we achieved better quality at a even lower cost. Our proposal provides a one-shot processing, while LARK involves multiple iterations without providing a terminating strategy. On the other hand, our proposal only does analysis with in a bounded local window, avoided the potentially expensive non-local processing, such as the patch clustering in BM3D.

**Visual Comparison**

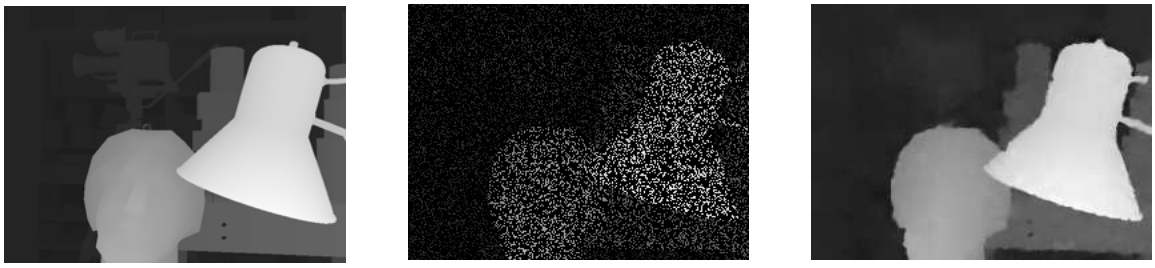
In this subsection, we show the visual comparison of the images. As we can see from the images shown below. Our proposed image provides sharper edges and smoother surfaces comparing to the other two competing schemes.



(a) Ground truth (b) Sensed Signal (c) Proposed

(b) LARK (c) BM3D

Figure 4-2: Ground truth, sensed signal and the visual comparison between LARK, BM3D and our proposal for sequence New Tsukuba 1.



(a) Ground truth (b) Sensed signal (c) Proposed

(d) LARK (e) BM3D

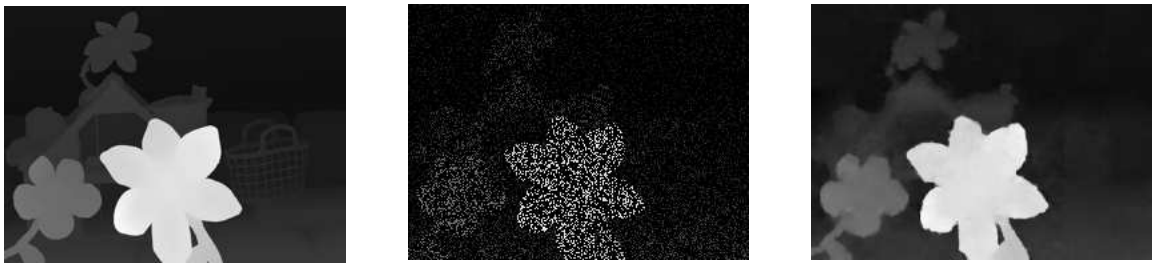
Figure 4-3: Ground truth, sensed signal and the visual comparison between LARK, BM3D and our proposal for sequence New Tsukuba 3.

### 4.7.2 $z$-dimensional DIBR with Interpolated Depth Images

#### Experiment Setup

Further, we did comparison by performing $z$-dimensional DIBR using the depth image restored from partial noising sample by different algorithms. In accordance with the experiments in Chapter 3, we assume the captured image $v_0$ and depth image $d_0$ are observed from a *near-camera*, and synthesize the images $v'_r$ and $d'_r$ as observed from the *far-camera* via DIBR. Next, by adding AWGN and removing the 75% of the pixels to simulate the sparsely sensed depth image $d^s_r$. Then, image recovering algorithm is applied to create an estimation $\hat{d}_r$ of the depth map. Finally, using $v'_r$ and $\hat{d}_r$ as the images from reference view, we synthesized back the near-camera image $\hat{v}_0$ and evaluate the synthesize qualities of $\hat{v}_0$.

We compare among different technique combinations. Especially, we compare the image synthesized from the combination of the proposed algorithms in Chapter. 3 and 4 versus the following two comparison schemes: a) depth restoration via LARK and DIBR using VSRS+, and b) depth restoration by classical gaussian



(a) Ground truth (b) Sensed signal (c) Proposed

(d) LARK (e) BM3D

Figure 4-4: Ground truth, sensed signal and the visual comparison between LARK, BM3D and our proposal for sequence Flowers.

kernel regression and DIBR using VSRS+. We used the images from New Tsukuba sequence and the middlebury stereo data set.

**Experiment Results**

The numerical results are shown in the following table:

As we can see from Table. 4.2, Our proposal outperforms the other two methods by up to 5.28dB and 7.39dB in PSNR respectively, and also ranks highest in SSIM value, which showed the effectiveness of combined application of our proposed algorithms.

## 4.8 Chapter Summary

Sparse sensing technique has been proven to be effective in reducing energy consumption by depth sensing. In order to use the sparse sensed depth images for freeview image / video applications, I proposeda joint denoise-interpolation scheme for depth images. Specifically, I proposed the piecewise-linear plus residual im-

Table 4.2: PSNR and SSIM comparison of synthesized images using different techniques

| | PSNR | | | SSIM | | |
|---|---|---|---|---|---|---|
| | Proposed | LARK & VSRS+ | KR & VSRS+ | Proposed | LARK & VSRS+ | KR & VSRS+ |
| New Tsukuba 1 | **30.12** | 25.05 | 27.53 | **0.9210** | 0.9063 | 0.9035 |
| New Tsukuba 2 | **29.68** | 24.79 | 27.23 | **0.9049** | 0.8805 | 0.8917 |
| New Tsukuba 3 | **31.16** | 27.28 | 27.82 | **0.9134** | 0.9082 | 0.9118 |
| Laundary | **30.90** | 27.04 | 25.37 | **0.9315** | 0.9158 | 0.9253 |
| Flowers | **26.53** | 24.28 | 25.09 | **0.8511** | 0.8362 | 0.84s01 |
| Art | **28.51** | 23.23 | 20.12 | **0.8934** | 0.8251 | 0.8104 |

age model for depth images. Further, the residuals are regularized via the graph Laplacian prior via a densely constructed graph. The experiments showed that our proposed scheme can significantly outperform the state-of-art schemes LARK and BM3D for 1.26dB and 1.18dB, respectively. Further, we evaluated the quality of DIBR synthesized images using the resonstructed depth maps, the results show that our proposal can restore the depth map that leads to better DIBR synthesis quality.

# Chapter 5

# Conclusion and Future Work

## 5.1 Main Contributions

Freeview image / video technology had been very popular and received much attention from both academia and industry, for the reason of its abundant potential applications. While the quick technology advances in hardware had greatly improved the technology from different aspects, including compression, streaming, depth sensing, image rendering, its current status is far from satisfying.

In this thesis, improvements of the freeview image / video technology via image interpolation and denoising by GSP are proposed. Especially, from the content consumer side, we proposed the $z$-dimensional extension of the system. The GSP-based image denoising and interpolation method is applied to fill the expansion holes – a new type of missing pixels occurred during a far-to-near viewpoint synthesis. Especially, we propose to first identify expansion holes via a depth layering procedure, then formulate a maximum a posteriori problem to estimate the missing pixels using a graph-signal smoothness prior. Then we propose an iterative reweighted least square algorithm to solve the proposed MAP problem efficiently. Experiment results show that visually pleasing images for a $z$-dimensional view switch can be successfully synthesized by our proposal. Further, we also used GSP-based image interpolation and denoising technique to restore depth image from partial samples. This allows the recent commodity depth sensors on portable

devices to work in a compressed sensing strategy, which reduces the energy consumption. We first presented a quadtree-decomposition based image model for depth images and then applied GSP to reconstruct the depth images. In the experiment, we achieved significantly better performance over the state-of-art BM3D and LARK schemes. Finally, experiment result proved that the combined usage of the proposed algorithms is proved to have better image reconstruction qualities for DIBR applications compared to traditional solutions.

## 5.2 Discussion

There are still many works to be completed to promote the usage of the proposed techniques. First, a key issue is the extension of our work from images to videos While we focus on static images to build the foundations, our work need to be extended to multiview video to draw interest from outside of academia. To extend our work to freeview videos, we need consider additional requirements such as temporal consistency [7, 59]. Second, while the depth restoration takes the samples from sensors as input, it is also able to tell the sensors which part of the scene is more important– we need more samples on edges and corners than on smooth surfaces. Instead of simply treat the depth image sensing and restoration as separated stages, we can have them work jointly, i.e. guide the sensor to take more samples in more important areas such as edges and corners, to generate more accurate depth maps. Further, in our discussion about the $z$-dimensional image synthesis work, we are using very accurate depth maps from the laboratories, which is not very likely to be available in real life applications. The noise model of the DIBR synthesized images using sensed depth maps, especially compressed-sensed and restored depth maps, could be very different and needs more research. Also, we can formulate the depth map restoration problem using the quality of the DIBR synthesized images as the criteria, instead of the depth map itself, for the purpose of a more practical evaluation. Finally, while in this thesis only the acquisition and the synthesis stage of the freeview video technology is discussed, the corresponding requirements on

compression and streaming still needs to be discussed.

## 5.3 Perspectives and Future Directions

Besides all the technical efforts to enhance freeview video technology, we will need more interesting applications to accelerate the popularization of freeview video technology. I'm optimistic about the future of free viewpoint technology not only because of its own potential, but also the rapid advancement of other related technologies, especially in the Virtual Reality field. Though the free viewpoint technology is originally developed for conventional 2D displays, I believe its ultimate media application is Virtual Reality (VR). While currently the VR equipment are mostly used to display the contents produced by computer graphics techniques, it can also be used to playback freeview videos, if we use the video from two viewpoints that moves correspondingly as the input. By using the freeview video technology as part of the VR implementation, we can expect a natural and fully-immersive experience, which can hardly be achieved with the current prevailing VR products merely based on Computer Graphics technologies.

# Bibliography


[1] P. Merkle, A. Smolic, K. Mueller, and T. Wiegand. Multi-view video plus depth representation and coding. In *IEEE International Conference on Image Processing*, San Antonio, TX, October 2007.

[2] W. Mark, L. McMillan, and G. Bishop. Post-rendering 3D warping. In *Symposium on Interactive 3D Graphics*, New York, NY, April 1997.

[3] I. Daribo and B. Pesquet-Popescu. Depth-aided image inpainting for novel view synthesis. In *IEEE Multimedia Signal Processing Workshop*, Saint-Malo, France, October 2010.

[4] I. Ahn and C. Kim. Depth-based disocclusion filling for virtual view synthesis. In *IEEE International Conference on Multimedia and Expo*, Melbourne, Australia, July 2012.

[5] S. Reel, G. Cheung, P. Wong, and L. Dooley. Joint texture-depth pixel inpainting of disocclusion holes in virtual view synthesis. In *APSIPA ASC*, Kaohsiung, Taiwan, October 2013.

[6] D. Tian, P.-L. Lai, P. Lopez, and C. Gomila. View synthesis techniques for 3D video. In *Applications of Digital Image Processing XXXII, Proceedings of the SPIE*, volume 7443 (2009), pages 74430T–74430T–11, 2009.

[7] T. Maugey, P. Frossard, and G. Cheung. Temporal and view constancy in an interactive multiview streaming system. In *IEEE International Conference on Image Processing*, Orlando, FL, September 2012.

[8] I. Daribo, G. Cheung, T. Maugey, and P. Frossard. RD optimized auxiliary information for inpainting-based view synthesis. In *3DTV-Conference*, Zurich, Switzerland, October 2012.

[9] M. Tanimoto, M. P. Tehrani, T. Fujii, and T. Yendo. Free-viewpoint TV. In *IEEE Signal Processing Magazine*, volume 28, no.1, January 2011.

[10] D. Farin, R. Peerlings, and P. H. N. de With. Depth-image representation employing meshes for intermediate-view rendering and coding. In *3DTV-Con*, Kos Island, Greece, May 2007.

[11] D. Scharstein and C. Pal. Learning conditional random fields for stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, MN, June 2007.

[12] A. Vetro and T. Wiegandand G. J. Sullivan. Overview of the stereo and multiview video coding extensions of the H.264 / MPEG-4 AVC standard. In *Proceedings of the IEEE*, volume 99, no.4, pages 626–642, 2011.

[13] K. Muller, H. Schwarz, D. Marpe, C. Bartnik, and et al. 3d high-efficiency video coding for multi-view video and depth data. In *IEEE Transactions on Image Processing*, volume 22, no.9, September 2013.

[14] R. Hartley. Theory and practice of projective rectification. In *International Journal of Computer Vision*, volume 35, no.2, pages 115–127, November 1999.

[15] G. Cheung, A. Ortega, and N.-M. Cheung. Interactive streaming of stored multiview video using redundant frame structures. In *IEEE Transactions on Image Processing*, volume 20, no.3, pages 744–761, March 2011.

[16] X. Xiu, G. Cheung, and J. Liang. Delay-cognizant interactive multiview video with free viewpoint synthesis. In *IEEE Transactions on Multimedia*, volume 14, no.4, pages 1109–1126, August 2012.

[17] R. Krishnamurthy, B.-B. Chai, H. Tao, and S. Sethuraman. Compression and transmission of depth maps for image-based rendering. In *IEEE International Conference on Image Processing*, Thessaloniki, Greece, October 2001.

[18] D. Liu, X. Sun, F. Wu, S. Li, and Y.-Q. Zhang. Image compression with edge-based inpainting. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 17, no.10, pages 1273–1287, October 2007.

[19] Y. Morvan, D. Farin, and P. H. N. de With. Multiview depth-image compression using an extended H.264 encoder. In *Advanced Concepts for Intelligent Vision Systems, Lecture Notes in Computer Sciences*, volume 4678, pages 675–686, 2007.

[20] S. Shimizu, M. Kitahara, H. Kimata, K. Kamikura, and Y. Yashima. View scalable multiview coding using 3-D warping with depth map. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 17, no.11, pages 1485–1495, November 2007.

[21] A. M. Tekalp, E. Kurutepe, and M. R. Civanlar. 3DTV over IP: End-to-end streaming of multiview video. In *IEEE Signal Processing Magazine*, November 2007.

[22] G. Cheung, A. Ortega, and T. Sakamoto. Coding structure optimization for interactive multiview streaming in virtual world observation. In *IEEE International Workshop on Multimedia Signal Processing*, Cairns, Queensland, Australia, October 2008.

[23] G. Cheung, A. Ortega, and N.-M. Cheung. Bandwidth-efficient interactive multiview video streaming using redundant frame structures. In *2009 APSIPA Annual Summit and Conference*, Sapporo, Japan, October 2009.

[24] H. Takeda, S. Farsiu, and P. Milanfar. Kernel regression for image processing and reconstruction. In *IEEE Trans. on Image Processing*, volume 16, no.3, pages 349–366, Feb. 2007.

[25] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. In *IEEE Signal Processing Magazine*, volume 30, no.3, pages 83–98, May 2013.

[26] F. K. Chung. *Spectral graph theory*. American Mathematical Society, 1997.

[27] W. Hu, X. Li, G. Cheung, and O. Au. Depth map denoising using graph-based transform and group sparsity. In *IEEE International Workshop on Multimedia Signal Processing*, Pula, Italy, October 2013.

[28] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovacevic. Signal denoising on graphs via graph filtering. In *IEEE Global Conference on Signal and Information Processing*, Austin, TX, December 2014.

[29] J. Pang, G. Cheung, A. Ortega, and O. Au. Optimal graph laplacian regularization for natural image denoising. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, Brisbane, Australia, April 2015.

[30] S. K. Narang, A. Gadde, and A. Ortega. Signal processing techniques for interpolation of graph structured data. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, Canada, May 2013.

[31] P. Wan, G. Cheung, D. Florencio, C. Zhang, and O. Au. Image bit-depth enhancement via maximum-a-posteriori estimation of graph ac component. In *IEEE International Conference on Image Processing*, Paris, France, October 2014.

[32] X. Liu, G. Cheung, X. Wu, and D. Zhao. Inter-block soft decoding of JPEG images with sparsity and graph-signal smoothness priors. In *IEEE International Conference on Image Processing*, Quebec City, Canada, September 2015.

[33] W. Hu, G. Cheung, A. Ortega, and O. Au. Multi-resolution graph fourier transform for compression of piecewise smooth images. In *IEEE Transactions on Image Processing*, volume 24, no.1, pages 419–433, January 2015.

[34] U. Luxburg. A tutorial on spectral clustering. In *Statistics and Computing*, volume 17, no.4, pages 395–416, December 2007.

[35] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: From error visibility to structural similarity. In *IEEE Transactions on Image Processing*, volume 13, no.4, pages 600–612, August 2005.

[36] C. Zhang, Z. Yin, and D. Florencio. Improving depth perception with motion parallax and its application in teleconferencing. In *IEEE International Workshop on Multimedia Signal Processing*, Rio de Jeneiro, Brazil, October 2009.

[37] S. Reichelt, R. Hausselr, G. Futterer, and N. Leister. Depth cues in human visual perception and their realization in 3D displays. In *SPIE Three-Dimensional Imaging, Visualization, and Display 2010*, Orlando, FL, April 2010.

[38] B. Macchiavello, C. Dorea, E. M. Hung, G. Cheung, and W. Tan. Loss-resilient coding of texture and depth for free-viewpoint video conferencin. In *IEEE Transactions on Multimedia*, volume 16, no.3, pages 711–725, April 2014.

[39] L. Toni, T. Maugey, and P. Frossard. Correlation-aware packet scheduling in multi-camera networks. In *IEEE Transactions on Multimedia*, volume 16, no.2, pages 496–509, February 2014.

[40] L. Toni, T. Maugey, and P. Frossard. Optimized packet scheduling in multiview video navigation systems. In *IEEE Transactions on Multimedia*, volume 17, no.8, pages 1604–1616, September 2015.

[41] D. Ren, G. Chan, G. Cheung, and P. Frossard. Coding structure and replication optimization for interactive multiview video streaming. In *IEEE Transactions on Multimedia*, volume 16, no.7, pages 1874–1887, November 2014.

[42] D. Ren, G. Chan, G. Cheung, V. Zhao, and P. Frossard. Anchor view allocation for collaborative free viewpoint video streaming. In *IEEE Transactions on Multimedia*, volume 17, no.3, pages 307–322, March 2015.

[43] Y. Mao, G. Cheung, A. Ortega, and Y. Ji. Expansion hole filling in depth-image-based rendering using graph-based interpolation. In *IEEE International Conference on Acousitics, Speech and Signal Processing*, Vancouver, Canada, May 2013.

[44] I. Daubechies, R. Devore, M. Fornasier, and S. Gunturk. Iteratively re-weighted least squares minimization for sparse recovery. In *Commun. Pure Appl. Math*, 2009.

[45] F. Battisti, E. Bosc, M. Carli, P. Le Callet, and S. Perugia. Objective image quality assessment of 3D synthesized views. In *Signal Processing: Image Communication*, volume 30, pages 78–88, January 2015.

[46] J. Bigun and G. Granlund. Optimal orientation detection of linear symmetry. In *Proceedings of the IEEE International Conference on Computer Vision*, london, UK, 1987.

[47] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proceedings of the IEEE International Conference on Computer Vision*, Bombay, India, 1998.

[48] Y. Mao, G. Cheung, and Y. Ji. Image interpolation during dibr view synthesis using graph fourier transform. In *3DTV-Conference 2014*, Budapest, Hungary, July 2014.

[49] A. Buades, B. Coll, and J. Morel. A non-local algorithm for image denoising. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, San Diego, CA, June 2005.

[50] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 13, no.7, pages 560–576, July 2003.

[51] H.S. Hou and H.C. Andrews. Cubic splines for image interpolation and digital filtering. In *IEEE Transactions on Acoustic, Speech, and Signal Processing*, volume 26, no.6, pages 508–517, January 1978.

[52] K. Bredies and M. Holler. A TGV-based framework for variational image decompression, zooming and reconstruction. part i: Analytics. In *SIAM Journal on Imaging Sciences*, volume 8, no.4, pages 2814–2850, 2015.

[53] Y. Romano, M. Protter, and M. Elad. Single image interpolation via adaptive nonlocal sparsity-based modeling. In *IEEE Transactions on Image Processing*, volume 23, no.7, pages 3085–3098, July 2014.

[54] X. Gao K. Zhang, D. Tao, and X. Li. Single image super-resolution with non-local means and steering kernel regression. In *IEEE Transactions on Image Processing*, volume 21, no.11, pages 4544–4556, November 2012.

[55] G. Yu, G. Sapiro, and S. Mallat. Solving inverse problems with piecewise linear estimators: From gaussian mixture models to structured sparsity. In *IEEE Transactions on Image Processing*, volume 21, no.5, pages 2481–2499, May 2012.

[56] H. K. Arachchi, S. Dogan, X. Shi, E. Ekmekcioglu, S. Worrall, A. Franck, C. Hartmann, T. Korn, and P. Kovacs. Distribution of multi-view entertainment using content aware delivery systems diomedes. Technical Report FP7-ICT-247996, European Commission, Octobor 2011.

[57] P. T. Kovacs, A. Barsi, A. Ouazan, B. Gunel, D. Doyen, U. Schreiber, D. Passoni, F. Zilly, A. Laborie, and G. Berenger. Multimedia scalable 3d for europe. Technical Report FP7-ICT-247010, Holografika, June 2011.

[58] A. Chuchvara, M. Georgiev, and A. Gotchev. Cpu-efficient free view synthesis based on depth layering. In *3DTV-Conference 2014*, Budapest, Hungary, July 2014.

[59] Z. Liu, G. Cheung, J. Chakareski, and Y. Ji. Multiple description coding and recovery of free viewpoint video for wireless multi-path streaming. In *IEEE Journal of Selected Topics in Signal Processing*, volume 9, no.1, pages 151–164, 2015.

[60] Y. Mao, G. Cheung, and Y. Ji. Graph-based interpolation for dibr-synthesized images with nonlocal means. In *Symposium on Graph Signal Processing in IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Austin, TX, December 2013.

# List of Publicated Journals

[1] Y. Mao, G. Cheung, and Y. Ji. On constructing z-dimensional dibr-synthesized images. In *IEEE Transactions on Multimedia*, 2016.

# List of Publicated Conference Papers


[1] Y. Mao, G. Cheung, C. Lin, and Y. Ji. Image classifier learning from noisy labels via generalized graph smoothness priors. In *IEEE IVMSP Workshop 2016*, Bordeaux, France, July 2016.

[2] C.Yang, Y. Mao, G. Cheung, V. Stankovic, and K. Chan. Graph-based depth video denoising and event detection for sleep monitoring. In *IEEE International Workshop on Multimedia Signal Processing 2014*, Jakarda, Indonesia, July 2014.

[3] Y. Mao, G. Cheung, and Y. Ji. Image interpolation during dibr view synthesis using graph fourier transform. In *3DTV-Conference 2014*, Budapest, Hungary, July 2014.

[4] Y. Mao, G. Cheung, and Y. Ji. Graph-based interpolation for dibr-synthesized images with nonlocal means. In *Symposium on Graph Signal Processing in IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Austin, TX, December 2013.

[5] Z.Liu, Y. Mao, N. Lu, Y. Ji, and S. Shen. Resource allocation for wwan video multicast with cooperative local repair. In *IEEE International Conference on Communications 2013*, Budapest, Hungary, July 2013.

[6] Y. Mao, G. Cheung, A. Ortega, and Y. Ji. Expansion hole filling in depth-image-based rendering using graph-based interpolation. In *IEEE International Conference on Acousitics, Speech and Signal Processing*, Vancouver, Canada, May 2013.

[7] Y. Mao, G. Cheung, and Y. Ji. Depth-layer-based multiview image synthesis & coding for interactive z- and x-dimension view switching. In *SPIE VIPC 2013*, Burlingame, CA, January 2013.