2015-07-09

Compressive Volume Rendering

Liu, Xiaoyang

Liu, X. (2015). Compressive Volume Rendering (Unpublished master's thesis). University of Calgary, Calgary, AB. doi:10.11575/PRISM/25391

http://hdl.handle.net/11023/2346

master thesis

UNIVERSITY OF CALGARY

Compressive Volume Rendering

by

Xiaoyang Liu

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN COMPUTER SCIENCE

CALGARY, ALBERTA

July, 2015

Abstract

Compressive rendering refers to the process of reconstructing a full image from a small subset of the rendered pixels, thereby expediting the rendering task. Images produced via direct volume rendering are usually highly compressible in a transform domain such as the Fourier or wavelet domains. In this dissertation, we empirically investigate four image order techniques for compressive rendering that are suitable for direct volume rendering. The first technique is based on the theory of compressed sensing and leverages the sparsity of the image gradient in the Fourier domain. Following this, we investigate sparse representation of volume rendered images via dictionary learning. The latter techniques exploit smoothness properties of the rendered image; the third technique recovers the missing pixels via a total variation minimization procedure while the fourth technique incorporates a smoothness prior in a variational reconstruction framework employing interpolating cubic B-splines. We compare and contrast these four techniques in terms of quality, efficiency and sensitivity to the distribution of pixels. Our results show that smoothness-based techniques significantly outperform techniques that are based on compressed sensing as well as dictionary learning and are also robust in the presence of highly incomplete information. We achieve high quality recovery with as little as 20% of the pixels distributed uniformly in screen space.

Acknowledgements

First and foremost I would like to thank my supervisor Dr. Usman R. Alim for introducing me to the field of scientific visualization. Thank you Dr. Usman for granting me this opportunity to join the Visualization and Graphics Group (VISAGG) and being a guiding force and a source of constant inspiration and professionalism for my research work.

Thank-you to all VISAGG members, especially Allan Rocha, for your extensive comments and help throughout my Masters' degree. I really enjoyed all the coffee breaks we took together and it has been so much fun to work with you guys.

I would also like to thank Dr. Leonard Manzara for giving me the opportunity to being a Teaching Assistant for your Data Structures and Algorithms in Java course. It has been a wonderful experience learning and tutoring students on interesting algorithms.

To my committee members, Dr. Sonny Chan and Dr. Kartikeya: Thank you so much for being my committee members and providing insightful and informative advice.

Lastly, I would like to thank my girl friend Wanting and my parents for the endless supports and being my source of energy to learn, explore and grow.

Table of Contents

List of Tables

List of Figures and Illustrations

Chapter 1

Introduction

This thesis is intended to address the research problem of pursuing efficient compressive volume rendering techniques. To begin with, we describe the context of our research and the problems faced by the researchers. Motivated by this compressive volume rendering objective, we introduce the methodologies we followed to achieve it. Last but not the least, we conclude the introduction chapter with our contributions and an organization of this thesis.

1.1 Background

The need to to efficiently process and analyze data sets is recognized by various disciplines. As large screen and high density displays are becoming commonplace, we are faced with the additional challenge of adapting our visualization algorithms so that they can produce better quality large size images. In the context of a ray-casting based approach to volume visualization, the total rendering time is usually proportional to the number of pixels in the rendered image which in turn depends on the number of rays cast per pixel. Sophisticated illumination effects, high-quality data filtering and out-of-core data management techniques make the process even more expensive. It is therefore imperative to employ acceleration strategies that reduce the overall cost while maintaining image quality.

1.2 Volume Rendering

In scientific visualization and computer graphics, volume rendering is a set of techniques used to display a 2D projection of a 3D discretely sampled data set, typically a 3D scalar field [Wik15b]. Most volume rendering data sets are acquired by X-ray computed tomography

Figure 1.1: Head image generated via volume rendering

(CT), Magnetic resonance imaging (MRI) or other devices like MicroCT scanners. Those 2D slice images are composed of a regular number of image pixels. Voxel is the basic element of a volumetric data set and it represents a value on a regular grid in 3D space. An RGBA (for red, green, blue, alpha) transfer function is usually used to define the RGBA components for each voxel value. This color and opacity information is of vital importance to specify the rendering parameters for volume rendering tasks. Transfer functions make volume data visible by mapping data values to optical properties. Transfer functions are also related to the smoothness of the rendered image. The rendered images can be very smooth if the transfer function is changing smoothly without abrupt jumps between neighbouring values. In contrast, the smoothness of output images can be significantly low if the transfer functions are designed for iso-surface rendering [BMWM06]. Fig. 1.1 is one example of volume rendering obtained with a relatively smooth transfer function.

Ray casting is commonly used for direct volume rendering since it produces high quality

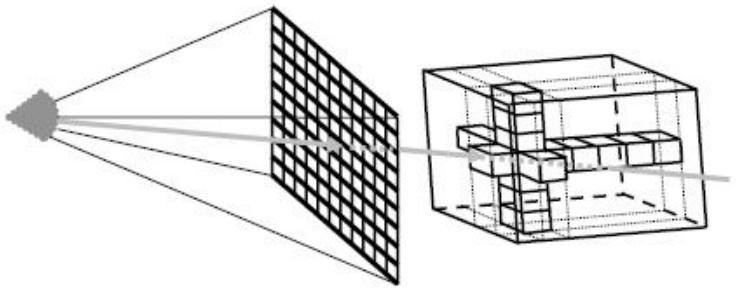


Figure 1.2: Volumetric ray casting

rendering results. It is an image-based technique that computes pixels for 2D images from voxels of 3D volumetric data sets. Fig. 1.2 illustrates the big picture of volume rendering via ray casting. It is composed of the virtual camera, image plane and the 3D volumetric data set. Rays from the camera step through the volumetric data and sample the 3D data by capturing color and opacity information along the ray. In the end, the sampled data is projected to the 2D image plane which we perceive as the high quality image.

Basically, we can break ray casting into four main steps as Fig. 1.3 demonstrates:

1. **Ray casting** At this stage, the 3D volumetric data will be touched and it is sometimes useful to consider the volume being enclosed within a cubic box as illustrated in Fig. 1.2. Rays are pushed through the bounding box while intersecting with the volume which is of vital importance to the following sampling step.

2. **Sampling** Along the ray, sample points are selected equidistantly. The sample distance parameter has to be carefully tuned since it is of significant importance to the output image quality. The increment of number of samples leads to more accurate sampling, which makes the rendering time longer accordingly. At each sample point, some interpolations from neighbouring voxels are needed since the volume is not usually aligned with the direction of the ray. Trilinear interpolation is commonly used but higher order interpolation leads

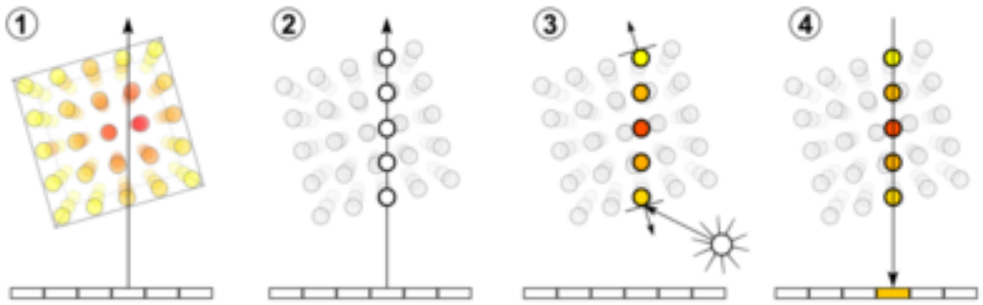


Figure 1.3: The four basic steps of volume ray casting: (1) Ray Casting (2) Sampling (3) Shading (4) Compositing.

to better image quality.

3. **Shading** At this stage, for visual realism, the interaction with light sources is considered. It is done by computing a gradient of scalar values for each sampling point followed by the shading process according to the relationships between the surface orientation as determined by the gradient and the location of the light sources in the scene.

4. **Compositing** All sampling points are composited along the ray after they have been shaded, producing the final color value for the current image pixel. With the help of the rendering equation, the compositing can be done either in back-to-front or front-to-back order. This compositing process can be considered as blending acetate sheets on an overhead projector.

Direct volume rendering can be computationally expensive due to the expensive stages in the pipeline. By using a direct volume renderer, one has to provide every voxel value that has to be mapped to color and opacity information via a transfer function. As the number of image pixels grows, high quality rendering is costly and slow.

Volume rendered images typically tend to be spatially slowly varying and lack sharp fluctuations such as textures. This is caused by the partial volume effect that occurs during data acquisition procedures of CT scanners. Partial volume effect is the loss of contrast

between two adjacent tissues in an image caused by insufficient resolution so that more than one tissue type occupies the same voxel (or pixel). It will, therefore, produce a beam attenuation proportional to the average value of these tissues. Accordingly, the generated 2D image slices tend to be smooth due to this effect.

Bearing this in mind, in this thesis, we explore different image-space methods to speed up the volume rendering process by using a subset of image pixels. Rather than rendering the volumetric data directly and compressing the rendering output afterwards, we obtain the rendering result with only a subset of pixels followed by a process that recovers the missing pixels. We are specially interested in good quality recovery and explore methods that yield good results with a small subset of the pixels as explained in the following section..

1.3 Methodologies

Recovering missing pixels is usually done by solving a linear system. Linear systems are quite common in computer graphics and come in different forms:

- **Overdetermined:** Too many equations with too few unknowns. Visually, this corresponds to a tall and thin matrix and can be solved via least squares by minimizing the sum of the squares of the errors made in the results of every single equation.
- **Square Matrix:** The simplest form which has the same amount of equations and unknowns.
- **Underdetermined:** In contrast, this system has too few equations and too many unknowns, which is a fat and short matrix. This is what compressed sensing deals with.

In the context of compressive volume rendering, we intend to recover volume rendered images from a small fraction of pixels. In this case, the measurement vector is much shorter

than the original image vector we are trying to estimate. Therefore, the system we are dealing with is an underdetermined one. With an infinite number of solutions, it is therefore necessary to incorporate some a priori information about the solution in order to recover an image that is most consistent with the measured pixels. Sen and Darabi [SD11] exploited the fact that rendered images are sparse in the wavelet domain and reported good recovery results with 75% of the pixels.

We build upon the idea of compressive rendering specifically in the context of volume rendering, where we postulate that images, owing to their inherent smoothness, should be recoverable from a much smaller fraction of the pixels. Towards this end, we explore recent techniques inspired from the image-processing community. While these techniques can be viewed from different angles (image restoration or completion, compression etc.), our interest is in studying them from the point of view of compressive volume rendering. It is unclear which method is the most suitable for this purpose, specially when the fraction of rendered pixels is very small as compared to the total number of pixels. This research work aims to definitively answer this question so as to guide researchers and practitioners in designing more efficient volume rendering strategies.

1.3.1 CS(Compress Sensing)-based Approaches

Our first technique is an extension of the idea of Sen and Darabi [SD11]. Instead of rendering the original image, we render a subset of the gradient of the image and exploit the sparsity of the discrete Fourier transform of the gradient components to recover a gradient image that is most consistent (in the least squares sense) with the measurements. In mathematics, the discrete Fourier transform (DFT) converts a finite list of equally spaced samples of a function into the list of coefficients of a finite combination of complex sinusoids, ordered by their frequencies, that has those same sample values. Thus, it can be taken as a tool which transforms one sampled function from its original spatial domain to the frequency domain. Operations in the frequency domain (i.e. on the Fourier transform) are of significant impor-

tance to the digital image processing community. For data compression, the transformed Fourier coefficients represent the importance of those signal segments. High frequencies in the signal are taken as details and assumed to be unnoticeable thereby discarded. By doing this, most important information of the image can be retained while other insignificant ones are discarded, which lead to a compressed version of the original image. It is the case for images that are composed of large flat contents, such as sky and clouds. High frequencies need to be carefully examined if the most important parts of one image are corners and edges.

With those approximated gradient components, we use a Poisson solver to recover the original image. Unlike the approach of Sen and Darabi [SD11], this approach does not suffer from coherence issues since the Fourier basis is inherently incoherent with the canonical sampling basis. Moreover, the gradient components of a volume rendered image are typically more sparse in the Fourier domain as compared to the image itself. Our results show an improvement over the results of Sen and Darabi [SD11]. However, both methods quickly deteriorate when the fraction of missing pixels is high.

1.3.2 Dictionary Learning

We also explore overcomplete dictionaries (both preconstructed and learned) in the context of volume rendering. In particular, we are interested in investigating the suitability of such dictionaries for the purpose of sparsely representing volume rendered images. There are potentially many uses for such dictionaries in the volume rendering context. Besides the obvious relationship between sparsity and compression, such dictionaries are also potential candidates for compressive rendering (or inpainting in image processing parlance). We apply the theory of sparse and redundant representations to volume rendered images. In particular, we investigate dictionaries trained via one dictionary learning algorithm. We present empirical qualitative and quantitative results that demonstrate how well these dictionaries perform with respect to compressive rendering as well as typical volume exploration tasks

such as changes in rendering parameters, transfer function, or viewpoint. We also explore the generalizability of trained dictionaries to arbitrary volumetric datasets.

1.3.3 Smoothness-based Approaches

In order to guarantee robustness to highly incomplete information, we explore methods that are fundamentally different from CS and employ smoothness instead of sparsity priors. This is a more natural setting for this problem as smoothness is a key property of volume rendered images. Towards this end, we explore two methods namely total variation (TV) minimization and, variational minimization of a smoothness norm in a spline space (SS). Both of these methods predate CS but have not been explored in the context of compressive rendering. As the name suggests, TV minimization attempts to minimize the total variation norm that is intimately associated with the smoothness of an image [NW13]. The latter method seeks to find a smooth solution in a shift-invariant spline representation of the image. This is achieved by minimizing a least-squares type norm that penalizes non-smooth solutions [ASHU05]. We show that these smoothness inspired methods are much more resilient to missing information as compared to the CS-based methods. Moreover, they also have a performance advantage since the minimization process is less expensive as compared to the pursuit strategies typically employed in CS.

1.4 Contributions

This research work provides novel contributions to the state of the art in compressive volume rendering techniques.

We investigate four different methods for recovering images from a subset of the pixels, which in return, speed up the volume rendering process while maintaining great image quality.

We empirically show that compressed sensing based approaches are not suitable for this

problem and show their limitations on performing compressive volume rendering tasks.

We study sparse representations and investigate the use of over-complete dictionaries for representing volume rendered images. We show its limitations on recovery quality when the percentage of missing pixels is high while presenting its advantage in terms of high level of sparsity.

We propose two smoothness-based methods and show that they are more suitable for recovering images from a tiny fraction of pixels.

By comparing and contrasting recovery results via different methods, we manage to answer the question of which method is the most suitable one for compressive volume rendering. Hopefully, this will guide researchers and practitioners in designing more efficient volume rendering strategies.

Our poster "Compressive Volume Rendering in the Gradient Domain" was accepted by EuroVis 2014 which was held in Swansea, UK. Moreover, this poster got second place prize in the CPSC industry day student poster competition and we were invited to present this poster work during the event. Following this poster, our paper "Compressive Volume Rendering" was unconditionally accepted by EuroVis 2015 and we were invited to present our work during the conference in Cagliari, Italy in May 2015.

1.5 Thesis Organization

The remainder of the document is organized as follows. After reviewing relevant prior art, we present a detailed description of our proposed methods. These methods are then thoroughly compared and contrasted in terms of image quality, sensitivity to the distribution of pixels, and performance. Even though we focus on image quality in the volume rendering setting, we show that smoothness is also relevant in the general rendering context and leads to significantly better recovery as compared to the state of the art.

Chapter 2

Related Work

This chapter briefly introduces our readers to the existing technologies, theories and systems that have enlightened our research.

We briefly describe sparsity and its close relationship with Compressed Sensing theory as well as dictionary learning techniques. In addition, we discuss the topic of exploiting sparsity in well-known transform domains such as the discrete cosine transform and the discrete wavelet transform. It lays the foundations for our CS-Gradient method.

Next, we discuss some existing image processing methods for missing data recovery, such as inpainting. These techniques inspired us to further investigate image-order recovery methods for reconstructing a full image from a small subset of the rendered pixels.

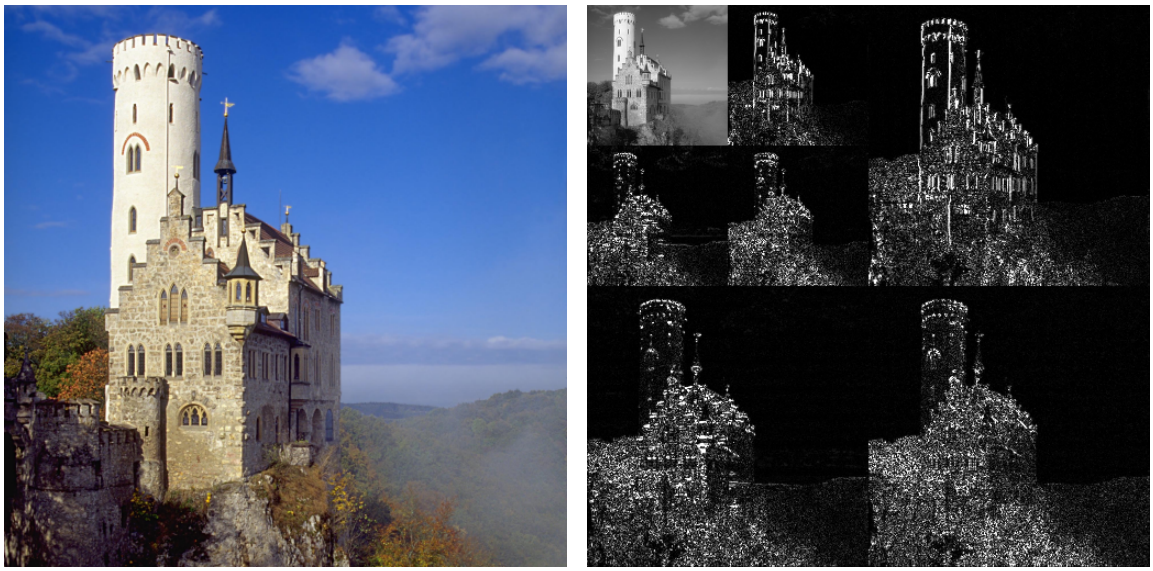
We also review other state of the art approaches to accelerate the volume rendering process.

2.1 Sparsity

In terms of data representations, sparsity and density are terms used to describe the percentage of cells in a database table that are not populated and populated, respectively. The sum of the sparsity and density should equal 100%.

A vector or matrix is sparse when it has few non-zero entries. Sparsity can lead to significant accelerations for matrix operations such as matrix multiplication. Specifically, it can transform matrix multiplication of two $N \times N$ matrices, normally a $O(N^3)$ operation, into an $O(k)$ operation, where k is the number of non-zero elements in the matrix.

From the image point of view, sparsity means similarity of image content in some domain. For example, natural images are largely the same color in areas (the sky is blue while the



(a) Original image (b) Wavelet transform of a natural image

Figure 2.1: An example of the 2D discrete wavelet transform [Wik15c] that is used in JPEG2000 [TM]

grass is green, etc). In our research work, we are inspired by the fact that volume rendered images are typically spatially slowly varying and lack sharp fluctuations such as textures. This inherent smoothness guides us to exploit the sparsity of the image in a transform domain such as the Fourier or wavelet domain.

Fig. 2.1 illustrates the wavelet transform of a natural image. The output is sparse through the recursive nature of the wavelet. The original image has both flat surfaces, sharp edges and details. It has a lot of details in the bottom left while the sky has almost no details. Moreover, the edges of the castle are very sharp, so they can be clearly seen in the wavelet transform. Fig. 1.1 is one example of a volume rendered image we used in our experiments and it is easy to reckon that this image is spatially slow varying since the similarity of image content is relatively high.

2.1.1 Compressed Sensing

Compressed sensing is the way of acquiring signals with the prior knowledge that the signals of interest are sparse [Ela10]. As we discussed in Section 1.3, an underdetermined system

has too few equations and too many unknowns, which is a fat and short matrix. Compressed sensing can be taken as a linear algebra statement which can find a solution to these underdetermined linear systems of equations on the condition that we have the prior knowledge that the unknown solution is sparse.

In the Graphics and Visualization communities, interest in exploiting transform domain sparsity is gaining momentum and started with applications of compressed sensing to light transport [PML*09, SD09]. More recent works have explored these ideas in the context of sparsely representing volumetric datasets [WAG*12, GIGM12, XSE14].

2.1.2 Sparse Representation and Dictionary Learning

In clustering, a set of exemplar vectors is clustered and represented by some cluster centers (atoms) based on the Euclidean distance. In K-Means clustering, the set of cluster centers can be represented as $\{\mathbf{d}_k\}_{k=1}^{K}$. This can be considered as an extreme sparse representation where only one atom in used in the signal decomposition and the corresponding coefficient is one. In other words, the coefficient vector x_i contains only one nonzero coefficient which is one and all the other $n-1$ coefficients are zero. Obviously, this is an extreme case of sparse representation. In the more general case, the signal is represented as a linear combination of these dictionary atoms $\{\mathbf{d}_k\}_{k=1}^{K}$. Thus, sparse representation can be regarded as a generalization of the clustering problem.

There are various training algorithms that can all be regarded as generalizations of K-Means. The differences lie in the method used to obtain the sparse coefficients and the method used to iteratively update the dictionary atoms. These algorithms can be summarized as *Maximum Likelihood Methods*, *Method Of Optimal directions* (MOD) [EAH00] and *Maximum A-Posteriori Probability Approach* [KDMR*03, MKD01] as well as *Unions of Orthonormal Bases* [LGBB05]. In this work, we employ the K-SVD dictionary training algorithm [MAB06]. It is a simple and flexible algorithm that has recently found widespread use in image processing. It is also a generalization of K-Means that alternates between a

sparse coding step and a dictionary update step. The coding step uses a pursuit algorithm (such as *orthogonal matching pursuit*(OMP) [CBL89, DMZ94, PRK93, Tro04]) to seek sparse representations of a set of training vectors based on the current dictionary. The dictionary update step then updates the dictionary atoms to better fit the training vectors.

Compression and level of detail strategies in volume visualization have largely focused on handling volumetric datasets. GPU based methods are the state of the art and are surveyed in the recent paper by Rodríguez *et al.* [RGG*12]. A closely related work is that of Gobbetti *et al.* [GIGM12]. It employs the K-SVD algorithm while applying the concept of coresets [CS07, AHPV05] to process extremely massive volumes off-line and supports quick GPU-accelerated real-time reconstruction. Our approach is similar in that we also make use of the K-SVD algorithm without using coresets. However, we focus on the image domain where the dictionary training phase has to be adapted to handle color channels. Unlike the work of Sen and Darabi [SD11], overcomplete dictionaries are inherently coherent and recovery via a pursuit strategy — even in the presence of incomplete information — does not rely on incoherence. Furthermore, our dictionary training phase employs all three colour channels in order to learn the possible correlations between the channels.

2.1.3 Other Approaches

Transform domain strategies are no stranger to graphics and visualization. There are a multitude of approaches that exploit compressibility in well-known transform domains such as the discrete cosine transform and the discrete wavelet transform.

There are some other lesser known transforms such as shearlets [GL07] and curvelets [MP10] that are better able to describe anisotropic features such as edges. However, in order to use these in a compressive sensing framework, one needs to employ a sampling basis that is incoherent with these transform basis. Since compressive rendering makes pixel measurements (corresponding to the canonical basis), the choice of transform domain is constrained to the discrete Fourier or cosine transforms.

Besides the approaches presented in this thesis, there are other recent approaches to missing data recovery that are not considered in this thesis and are a subject of future work. Examples include matrix completion [CR09], and tensor completion [LMWY13]. Some classical approaches to the problem of missing data recovery include radial basis functions [Buh00] and inpainting [BSCB00].

2.2 Image Inpainting

Our research work is fundamentally an image-order acceleration approach which is inspired from the image-processing community. In this section, we briefly describe some recent image processing techniques that motivate us throughout our research work.

Inpainting [Wik15a] denotes the process of reconstructing lost or damaged parts of images and videos. In the real world, in the case of restoring a valuable painting in the museum, it can be done by a skilled art conservator or art restorer. In the digital world, inpainting refers to the process of reverting deterioration, adding or removing elements of the image data via sophisticated algorithms. Fig. 2.2 demonstrates one application of this technique. The original image is damaged near the top of this man's hair in addition to the cracks near the bottom. The challenge lies in maintaining the continuity of the texture of man's shirt as well as his hair.

The work of Bertalmio *et al.* [BSCB00] presents a novel algorithm for automatic digital inpainting, which does not require any user intervention once the region to be inpainted has been specified. Similar to inpainting, our dictionary learning method can be applied to recover image with missing pixels by exploiting sparsity of the image. This missing data recovery technique motivates us to further explore other smoothness based approaches for compressively rendering volumetric data, such as TV (Section 3.3) and SS (Section 3.4) method.

Figure 2.2: Original and restored image

2.3 Other Acceleration Strategies

It is important to stress that our approach is fundamentally an image-order acceleration approach and can be used in conjunction with the plethora of object-order acceleration techniques that are available. GPU-based ray-casting is the state of the art approach to volume rendering.

Recent decades have witnessed an evolution of consumer graphics hardware from fixed-function architectures to fully programmable floating-point graphics processors with more than 100 million transistors [HKRs*06]. As computer graphics researchers, we can benefit from the introduction of true programmability of today's graphics processors. Interpolation and compositing are two major operations related to volume rendering and both operations can efficiently be performed on modern graphics hardware. Some high-quality real-time rendering techniques for volumetric data and effects, which harness the power of consumer graphics hardware, have been thoroughly explored, specially for GPU ray casting [HKRs*06]. Isosurface rendering via a ray-casting approach can be performed on consumer graphics hardware which can perform high-quality rendering at interactive speeds [HSS*05].

As described above, consumer graphics hardware has evolved at a breathtaking rate. The power of parallelism and programmability of today's GPUs makes it possible to perform compute-intensive operations such as ray-casting at interactive rates. However, the available

memory sizes of today's GPUs are not increasing at the same rate as the ever-increasing resolution and size of today's volume data. Recent efforts have focused on data management techniques that handle large datasets such that analysis of giga-, tera-, and petabytes of volume data can be performed on GPUs [BHP14]. Moreover, to overcome the challenges of long data transfer times as well as GPU memory size limitations, a variety of compression and level-of-detail data representation techniques for direct volume rendering (DVR) have been introduced [RGG*12].

In the broader context of Monte-Carlo rendering, adaptive image-order techniques have recently been used to reduce the number of rays traced [RKZ12]. These techniques usually employ multiple passes, incrementally improving imaging quality. In contrast, compressive rendering is a non-adaptive approach that only traces rays through a subset of the pixels. It can however be extended to a second pass where the placement of pixels is based on the results of the previous pass [SD11]. Our focus however, is on the non-adaptive first pass where we are interested in investigating the effect of the initial non-adaptive distribution of pixels on image quality.

Chapter 3

Methodologies

Let $\mathbf{x}$ denote the rendered image that is W pixels wide and H pixels high. For convenience, we treat $\mathbf{x}$ as an $N \times 1$ column vector, i.e. $\mathbf{x} = [x_1 \cdots x_N]^T$ where $N = WH$. We also restrict attention to scalar-valued images with the assumption that RGB images can be treated in an independent component-wise manner. Instead of rendering all the pixels in $\mathbf{x}$, we are interested in rendering a small subset of the pixels. The rendered pixels are given by

$$\mathbf{y} = \mathbf{S}\mathbf{x}, \tag{3.1}$$

where $\mathbf{y} = [y_1 \cdots y_M]^T$ is an $M \times 1$ (typically $M \ll N$) column vector and $\mathbf{S}$ is a $M \times N$ binary sampling matrix. Each row of $\mathbf{S}$ is zero everywhere except for the pixel location that is to be retained. The recovery goal is then to estimate the full image $\mathbf{x}$ from the rendered pixels $\mathbf{y}$. Since the number of rendered pixels is much smaller than the total size of the image, this problem is inherently ill-posed. Some prior assumption about $\mathbf{x}$ needs to be incorporated in order to make the recovery process work. Table 3.1 summarizes the priors used in the methods presented in this thesis.

Method	**Property**
CS-Wavelet	Based on compressed sensing, assumes $\mathbf{x}$ is sparse in the wavelet domain.
CS-Gradient	Based on compressed sensing, leverages the sparsity of the gradient of $\mathbf{x}$ in the Fourier domain.
Dictionary Learning	Apply the theory of sparse and redundant representations to volume rendered images.
TV	Assumes that the $\mathbf{x}$ exhibits low total variation.
SS	Incorporates a smoothness norm based on the second derivatives of $\mathbf{x}$.

Table 3.1: Summary of different methods, CS-Wavelet is the work of Sen and Darabi [SD11] which is the state of the art in compressive rendering. We proposed four additional methods to extend the state of the art.

3.1 Gradient Recovery via CS

This approach is similar to the work of Sen and Darabi [SD11]. For the sake of comparison and completeness, we review briefly the theory of compressed sensing before proposing our solution that exploits sparsity of the gradient components in the Fourier domain. More details on compressed sensing can be found in the recent textbook [EK12].

3.1.1 CS Background

Let $\hat{\mathbf{x}} \in \mathbb{R}^N$ be a sparse vector, i.e. it has a few non-zero entries. Formally, sparsity is quantified by the ℓ_0-norm $\|\cdot\|_0$, that counts the number of non-zero entries. A vector $\hat{\mathbf{x}}$ is said to be k-sparse if $\|\hat{\mathbf{x}}\|_0 \leq k$. The sensing mechanism is modelled as a set of linear measurements that yield the vector $\mathbf{y} \in \mathbb{R}^M$. In particular,

$$\mathbf{y} = \mathbf{A}\hat{\mathbf{x}}, \tag{3.2}$$

where $\mathbf{A}$ is an $M \times N$ sensing matrix with $M \ll N$. Even though this system is underdetermined, it can be solved uniquely using compressed sensing as long as $\mathbf{A}$ meets the Restricted Isometry Condition (RIC):

$$(1-\delta)\,||\hat{\mathbf{x}}||_2^2 \leq \|\mathbf{A}\hat{\mathbf{x}}\|_2^2 \leq (1+\delta)\,||\hat{\mathbf{x}}||_2^2\,, \tag{3.3}$$

where $\delta \in (0,1)$ and $\|\cdot\|_2$ indicates the ℓ_2-norm. Intuitively, the RIC states that in a valid sensing measurement matrix $\mathbf{A}$, every possible set of k columns form an approximate orthogonal set. Matrices that have been proven (probabilistically) to meet RIC include partial Fourier or cosine matrices (randomly selected rows from the full discrete Fourier or cosine transform matrix), Gaussian and Bernoulli random matrices [CT06]. Under the condition of RIC, the corresponding recovery mechanism becomes non-linear and can be formulated as the optimization problem

$$\min ||\hat{\mathbf{x}}||_0 \quad \text{subject to} \quad \mathbf{A}\hat{\mathbf{x}} = \mathbf{y}. \tag{3.4}$$

This is an NP-hard problem. In practice, it is substituted by an ℓ_1-minimization problem which is solved using a pursuit algorithm [TG07]. Provided that the RIC is satisfied, recovery accuracy depends on the sparsity of $\hat{\mathbf{x}}$; various error bounds relating δ and k have been explored [EK12].

3.1.2 CS for Rendered Images (CS-Wavelet)

Usually, signals of interest such as rendered images are not inherently sparse, but are sparse in a transform domain such as the discrete wavelet transform (DWT) or the discrete cosine transform (DCT). In other words, the transformed representation $\hat{\mathbf{x}} = \Psi\mathbf{x}$ approximates the original image well with a few non-zero coefficients. Here, Ψ is an $N \times N$ orthonormal matrix corresponding to a compression basis. Substituting this transform domain representation into 3.1, we obtain the measurement equation

$$\mathbf{y} = \mathbf{S}\Psi^{-1}\hat{\mathbf{x}}. \tag{3.5}$$

Letting $\mathbf{A} = \mathbf{S}\Psi^{-1}$, it is clear that this equation corresponds to the sensing equation 3.2, and recovery is possible as long as the matrix $\mathbf{S}\Psi^{-1}$ satisfies the RIC. Verifying the RIC is computationally difficult and a useful related notion is that of coherence. The coherence of a sensing matrix $\mathbf{A}$, $\mu(\mathbf{A})$, is the largest absolute inner-product between any two columns $\mathbf{a}_i$ and $\mathbf{a}_j$:

$$\mu(\mathbf{A}) = \max_{1 \le i < j \le n} \frac{|\mathbf{a}_i^T \mathbf{a}_j|}{\|\mathbf{a}_i\|_2 \|\mathbf{a}_j\|_2}. \tag{3.6}$$

Intuitively, the lower the coherence, the better the sparse recovery via ℓ_1 minimization. When $M \ll N$, the coherence is lower bounded according to $\mu(\mathbf{A}) \ge 1/\sqrt{M}$.

Another way to look at coherence is in terms of the sampling matrix $\mathbf{S}$ and the compression matrix Ψ^{-1}. In order to guarantee the RIC, the two must be incoherent. The work of Sen and Darabi [SD11] exploits sparsity of the image in the wavelet domain. Unfortunately, the wavelet domain is *not* incoherent with point sampling. To improve incoherence, they recover a blurred version $\mathbf{x}_b$ of the original image $\mathbf{x}$, where $\mathbf{x}_b = \Phi\mathbf{x}$, and Φ is an $N \times N$ matrix

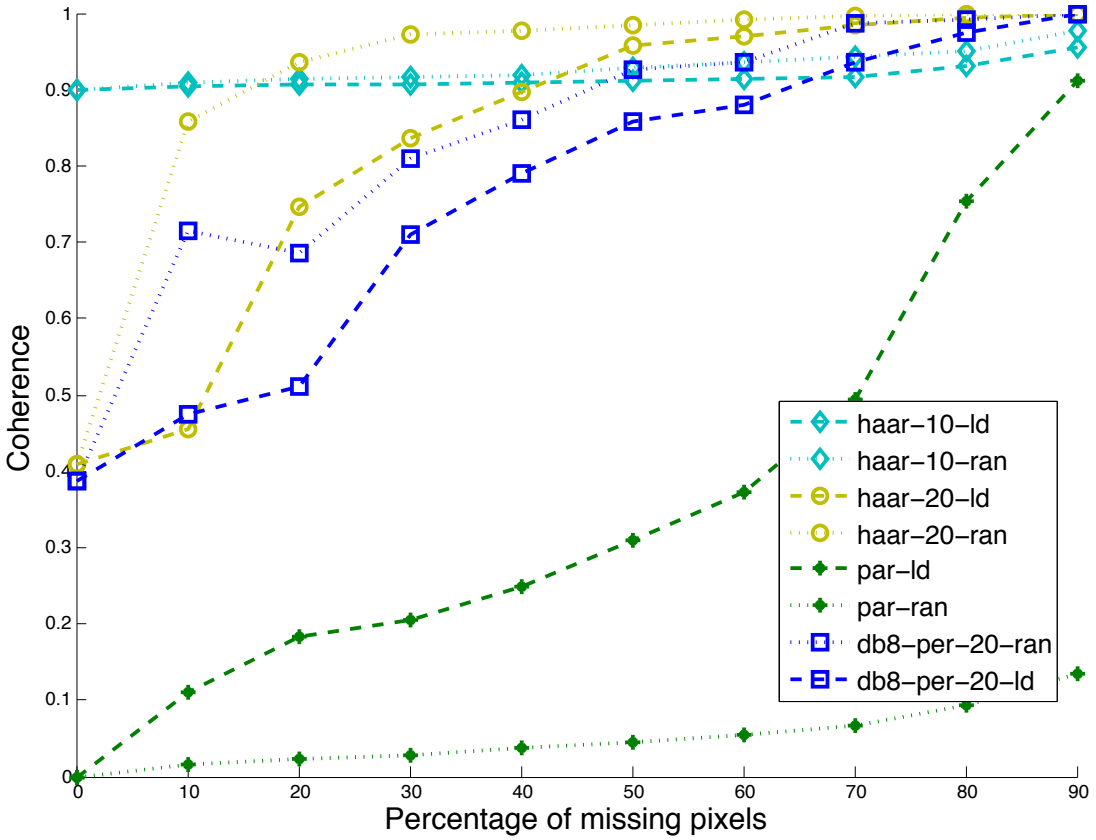


Figure 3.1: Coherence results for different sensing matrices ($N = 64^2$). The numbers indicate the standard deviation of the Gaussian blurring filter in the Fourier domain, lower values indicate greater blurring. The other abbreviations are: ld - low discrepancy, ran - random, and par - partial Fourier.

corresponding to the Gaussian blurring filter. Their sensing mechanism can be written as

$$\mathbf{y} = \underbrace{\mathbf{S}\Phi^{-1}\mathbf{W}^{-1}}_{\mathbf{A}}\hat{\mathbf{x}}_b, \tag{3.7}$$

where $\mathbf{W}^{-1}$ is the inverse DWT matrix, and $\hat{\mathbf{x}}_b$ is the DWT of the blurred image $\mathbf{x}_b$. From the recovered coefficients $\hat{\mathbf{x}}_b$, the final image is obtained according to $\mathbf{x} = \Phi^{-1}\mathbf{W}^{-1}\hat{\mathbf{x}}_b$. Even though, the blurring operation improves the incoherence somewhat, successful recovery is sensitive to the variance of the blurring filter Φ which needs to be adjusted on a case-by-case basis. As our tests show, this method is also very sensitive to the distribution of pixels and breaks down when the fraction of missing pixels is high. In the remainder of this thesis, we refer to the approach of Sen and Darabi [SD11] as CS-Wavelet.

3.1.3 CS-Gradient

In order to ameliorate coherence problems, we choose to exploit sparsity in the discrete Fourier transform (DFT) domain. The RIC property of random partial Fourier matrices is well-known. In other words, the Fourier domain is inherently incoherent with point sampling measurements. Fig. 3.1 compares the coherence of the sensing matrices in the CS-Wavelet method for a number of wavelet types as a function of the fraction of missing pixels. Observe that, for all wavelet types, coherence becomes higher as the fraction of missing pixels increases, and the blurring filter only improves incoherence slightly. On the other hand, random partial Fourier matrices exhibit very low coherence.

Rendered images are usually more sparse in the DWT domain as compared to the DFT domain [SD11]. In order to improve sparsity in the DFT domain, we can recover the image gradient rather than the image itself. For images that are slowly varying, we expect that the gradient components are more sparse in the DFT domain as compared to the image itself (Fig. 3.2). Observe that rendered images are also sparse in the gradient domain itself, and therefore, one can exploit sparsity in the gradient domain. However, the theory of CS dictates that one would need to make point measurements (of the image gradient components) in the DFT domain. This is suitable for applications such as Magnetic Resonance Imaging (MRI) [PMGC12], but cannot be realized in our case as the rendering process (barring applications such as frequency domain volume rendering [TL93]) typically makes pixel measurements.

Thus, instead of rendering the image $\mathbf{x}$ directly, we render the discrete gradient components $\mathbf{x}_1$ and $\mathbf{x}_2$. This can be done by representing $\mathbf{x}$ in a basis spanned by a tensor product (pixel reconstruction) kernel such as the bilinear B-spline or the Mitchell-Netravali cubic kernel [PH10]. The gradient can then be obtained by differentiating the kernel, i.e. instead of weighting the incoming rays with the pixel reconstruction filter, we can simply weight them according to the derivative of the kernel. Alternatively, one can use a box filter

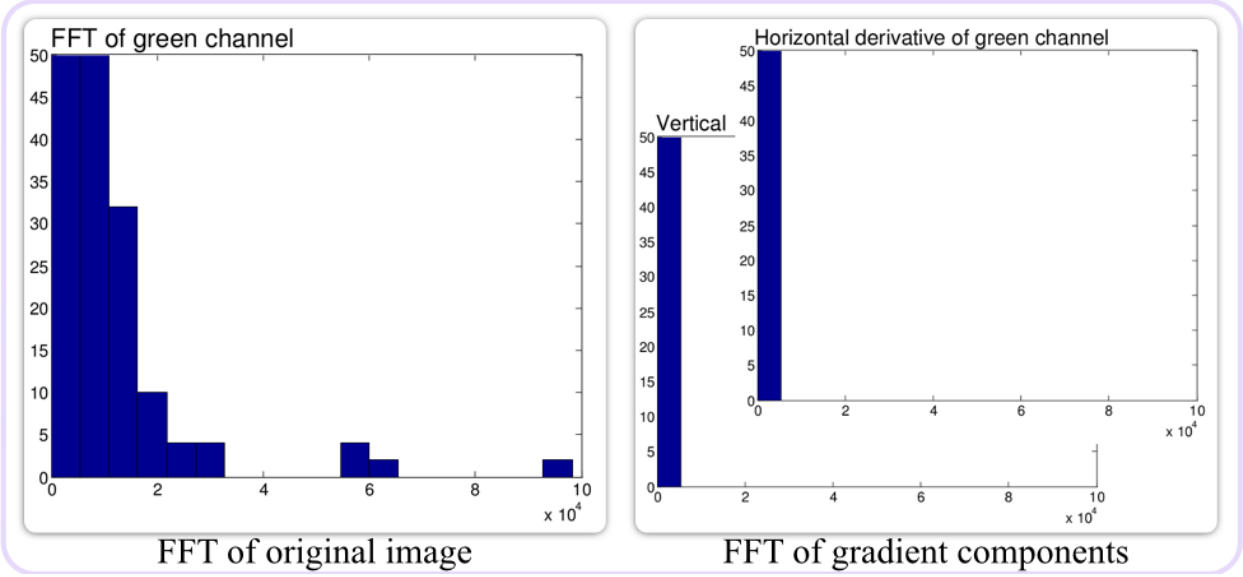


Figure 3.2: Histogram of the absolute values of the DFT coefficients of the engine image (left) and its derivatives (right). The image and the derivatives were normalized to lie in the range $[0, 1]$ before applying the FFT.

for rendering and apply a finite differencing scheme to obtain the gradient. Let $\mathbf{y}_1$ be an $m \times 1$ column vector that contains the horizontal component of the gradient measured at m different locations according to the sampling matrix $\mathbf{S}$. Our sensing mechanism can be formulated as

$$\mathbf{y}_1 = \mathbf{S}\mathbf{F}^{-1}\hat{\mathbf{x}}_1, \tag{3.8}$$

where $\mathbf{F}$ is the 2D DFT matrix, and $\hat{\mathbf{x}}_1$ denotes the 2D DFT of the horizontal gradient component $\mathbf{x}_1$. Observe that $\mathbf{A} = \mathbf{S}\mathbf{F}^{-1}$ is a partial Fourier matrix provided that the location of the pixels is random. An approximation of $\hat{\mathbf{x}}_1$ is obtained via the following ℓ_1 minimization:

$$\min \|\hat{\mathbf{x}}_1\|_1 \quad \text{subject to} \quad \|\mathbf{S}\mathbf{F}^{-1}\hat{\mathbf{x}}_1 - \mathbf{y}_1\|_2 \leq \epsilon. \tag{3.9}$$

An approximation of the DFT of the vertical component, $\hat{\mathbf{x}}_2$, is obtained similarly. Computing the pixel values from gradient components is a problem that frequently arises in gradient-domain image processing applications. We follow the approach of Pérez *et al.* and apply a Poisson solver—with Dirichlet boundary conditions—to the divergence of the gradi-

ent [PGB03].

3.2 Recovery via Dictionary Learning

In this section, we introduce some notions related to sparse representations and dictionary learning that will be relevant to our work. Readers requiring a more detailed treatment can refer to the book by Elad [Ela10] or to more recent surveys [RBE10, TF11].

3.2.1 Sparse Representation

An overcomplete dictionary is a collection of atoms (column vectors) such that the number of atoms exceeds the dimension of the signal space. Let $\mathbf{D} \in \mathbb{R}^{\mathbf{n}\times\mathbf{K}}$ ($n \ll K$) denote an overcomplete dictionary matrix that contains K prototype signal-atoms for columns $\{\mathbf{d}_j\}_{j=1}^{K}$. Given a signal $\mathbf{y} \in \mathbb{R}^{\mathbf{n}}$, one is interested in representing it as a sparse linear combination of the atoms in $\mathbf{D}$. The signal $\mathbf{y}$ can be either represented as $\mathbf{y} = \mathbf{D}\mathbf{x}$ or $\mathbf{y} \approx \mathbf{D}\mathbf{x}$, satisfying $\|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2 \leq \epsilon$, where ϵ is an error threshold. For sparse representation of signals, the goal is to find a solution $\mathbf{x} \in \mathbb{R}^{\mathbf{K}}$ that has the fewest number of nonzero coefficients. Thus, the sparsest representation is the solution of either

$$\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{y} = \mathbf{D}\mathbf{x} \tag{3.10}$$

or

$$\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2 \leq \epsilon \tag{3.11}$$

where $\|\cdot\|_0$ is the l_0 norm, counting the nonzero entries of a vector. There are many applications that benefit from sparsity and overcompleteness concepts (together or separately). Notably, the success of the JPEG2000 coding standard can be attributed to the sparsity of the wavelet coefficients of natural images [MGBB00].

3.2.2 Sparse Coding

This refers to finding a sparse solution $\mathbf{x}$ that has the fewest number of nonzero coefficients. Typically, one uses an approximate solution (3.11) instead of the exact sparsest representation (3.10) since extracting the exact sparsest representation is an NP-hard problem [DMA97]. This coding process can be completed by some pursuit algorithms that find an approximate solution. The simplest ones are the *matching pursuit* (MP) [MZ93] and the *orthogonal matching pursuit* (OMP) [CBL89, DMZ94, PRK93, Tro04] algorithms. They are similar in nature in that they are both greedy algorithms that select the dictionary atoms sequentially. OMP has some advantages over MP because of the improvement on orthogonality and faster convergence. The core idea is to compute the inner products between the signal and dictionary columns and possibly deploy some least squares solvers [MAB06]. Other pursuit approaches such as *basis pursuit* (BP) [CDS01] and its variation *focal underdetermined system solver* (FOCUSS) [GR97] are also possible. However, the sparsity of the coefficient matrix which leads to the successful sparsity evaluation is restricted by the properties of the dictionary $\mathbf{D}$. In this work, we focus on the OMP algorithm owing to its simplicity and relationship with the K-SVD dictionary learning algorithm.

3.2.3 Choice of the Dictionary

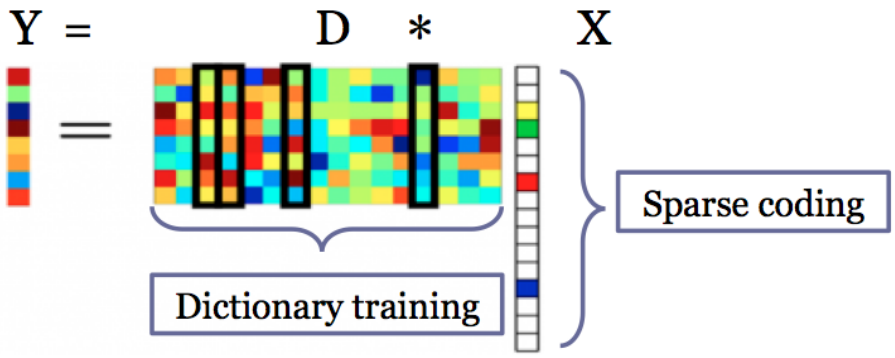


Figure 3.3: Sparse representation of one example signal $\mathbf{y}$, where $\mathbf{D}$ is the overcomplete dictionary and $\mathbf{x}$ contains the representation coefficients of this signal $\mathbf{y}$

As you can see in Fig. 3.3, we need a dictionary to represent signal $\mathbf{y}$ and many choices for overcomplete dictionaries are available. The dictionaries can either be preconstructed or learned.

Preconstructed dictionary: The dictionary can be chosen as a prespecified set of atoms such as the overcomplete discrete cosine transform (DCT), Haar or other overcomplete wavelet dicationaries. These generic dictionaries lead to simple and fast algorithms for the evaluation of the sparse representation (i.e. the product $\mathbf{Dx}$). On the other hand, they cannot ensure that the representation of a specific signal is sparse enough; their success depends on how suitable they are to sparsely describe the signal. Thus, they are only applicable to some specific kinds of images such as those that satisfy the shift-invariant property.

Learned Dictionary: Based on learning approaches, a dictionary that is more suitable to the task at hand can be designed. These kinds of dictionaries are applicable to a broader class of images and are potentially better than preconstructed dictionaries as they lead to lower sparsity levels. On the other hand, they are computationally expensive and are therefore restricted to some low-dimensional signals. In the context of images, this means that the images have to be broken upon into a set of patches, and each patch then forms the signal vector $\mathbf{y}$. However, with ever-growing computational capabilities, computational cost is not that significantly important compared to the improvement in sparsity.

3.2.4 Clustering and the K-SVD Algorithm

In clustering, a set of exemplar vectors is clustered and represented by some cluster centers (atoms) based on the Euclidean distance. In K-Means clustering, the set of cluster centers can be represented as $\{\mathbf{d}_k\}_{k=1}^{K}$. This can be considered as an extreme sparse representation where only one atom in used in the signal decomposition and the corresponding coefficient is one. In other words, the coefficient vector x_i contains only one nonzero coefficient which is one and all the other $n-1$ coefficients are zero. Obviously, this is an extreme case

of sparse representation. In the more general case, the signal is represented as a linear combination of these dictionary atoms $\{\mathbf{d}_k\}_{k=1}^K$. Thus, sparse representation can be regarded as a generalization of the clustering problem.

There are various training algorithms that can all be regarded as generalizations of K-Means. The differences lie in the method used to obtain the sparse coefficients and the method used to iteratively update the dictionary atoms. These algorithms can be summarized as *Maximum Likelihood Methods*, *Method Of Optimal directions* (MOD) [EAH00] and *Maximum A-Posteriori Probability Approach* [KDMR*03, MKD01] as well as *Unions of Orthonormal Bases* [LGBB05]. As discussed in Section 2.1.2, the K-SVD algorithm is a generalization of K-Means that alternates between a sparse coding step and a dictionary update step. The coding step uses a pursuit algorithm (such as OMP) to seek sparse representations of a set of training vectors based on the current dictionary. The dictionary update step then updates the dictionary atoms to better fit the training vectors. In particular, the algorithm solves the minimization problem:

$$\min_{\mathbf{D},\mathbf{X}} \left\{ \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 \right\} \quad \text{subject to} \quad \forall_i, \|\mathbf{x}_i\|_0 \leq L, \tag{3.12}$$

where $\mathbf{Y}$ is the matrix consisting of the training vectors $\mathbf{y}_i$, $\mathbf{X}$ is the matrix that contains the corresponding coefficient vectors $\mathbf{x}_i$, L is the target sparsity, and F stands for the Frobenius matrix norm. The algorithm converges under the condition that the sparse coding works perfectly which gives us the best approximation to signal $\mathbf{y}_i$ that contains no more than L nonzero entries. Additionally, each sparse coding step reduces the total representation mean squared error (MSE) $\|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2$ while the dictionary updating stage guarantees an additional reduction or no change in the MSE without violating the sparsity constraint. For a detailed discussion of the algorithm, we refer the reader to the original paper [MAB06].

With the trained dictionary available, we run experiments on images with missing pixels and Section 4.1.1 explains the detail of this procedure.

3.3 Recovery via TV minimization

The total variation seminorm of an image is defined as:

$$\|\mathbf{x}\|_{TV} := \sum_{i,j} |x_{i+1,j} - x_{i,j}| + |x_{i,j+1} - x_{i,j}|, \tag{3.13}$$

where $x_{i,j}$ — with slight abuse of notation — is the pixel value corresponding to the pixel with image-space coordinates (i, j). In other words, it is the sum of the ℓ_1 norms of the horizontal and vertical components of the gradient obtained via forward differencing. First proposed by Rudin *et. al* [ROF92], the TV-norm is low for images that are smooth and high for images that have high variability. It is also connected with sparsity; images that are sparse in the gradient domain also exhibit low TV. This property has been empirically known for some time and has been used in several applications such as denoising, inpainting, and recovery from partial Fourier measurements (see e.g [CEPY05]). The precise theoretical connection with CS has only recently been established [NW13].

Despite its success in image processing applications, TV minimization (or regularization) has not been used in the compressive rendering context which, at its core, is an image restoration problem akin to inpainting. Our goal here is to investigate how well this method performs as compared to the CS-based methods described earlier. The precise minimization problem that we wish to solve is given by

$$\min \|\mathbf{x}\|_{TV} \quad \text{subject to} \quad \|\mathbf{S}\mathbf{x} - \mathbf{y}\|_2 \leq \epsilon, \tag{3.14}$$

where $\mathbf{S}$ and $\mathbf{y}$ are as described in 3.1. This is a well-studied minimization problem and fast algorithms have recently gained popularity (e.g. split Bergman [GO09] or NESTA [BBC11]).

3.4 Recovery via Smoothness Splines (SS)

The fourth method we consider can be regarded as a scattered data approximation problem [XAE12]. It attempts to find a smooth solution in a prescribed space that is spanned by the uniform shifts of a kernel function $\varphi(\mathbf{t})$ where $\mathbf{t} \in \mathbb{R}^2$.

Let $\mathbf{j}_1, \ldots \mathbf{j}_N$ denote the pixel locations corresponding to the pixel values in $\mathbf{x}$, and let $\mathbf{k}_1 \ldots \mathbf{k}_M$ denote the pixel locations corresponding to the pixel values in $\mathbf{y}$. The goal is to find the coefficients $\mathbf{c} = [c_1 \cdots c_N]^T$ of the approximation:

$$f(\mathbf{t}) := \sum_{n=1}^{N} c_n \varphi(\mathbf{t} - \mathbf{j}_n), \tag{3.15}$$

such that the approximation closely matches the measured pixel values, i.e. $f(\mathbf{k}_m) \approx y_m$ for $m = 1, \ldots, M$. Additionally, it is desired that the function $f(\mathbf{t})$ be smooth. A useful notion of smoothness is provided by the second-order Beppo-Levi seminorm. For functions $g(\mathbf{t})$ and $h(\mathbf{t})$, the second-order Beppo-Levi inner-product is defined as

$$\langle g, h \rangle_{BL_2} := \langle \partial_{t_1 t_1} g, \partial_{t_1 t_1} h \rangle + 2 \langle \partial_{t_1 t_2} g, \partial_{t_1 t_2} h \rangle + \langle \partial_{t_2 t_2} g, \partial_{t_2 t_2} h \rangle \tag{3.16}$$

where $\langle \cdot, \cdot \rangle$ denotes the standard L_2 inner-product. The BL_2 inner-product induces a seminorm which we denote as $\|g\|^2_{BL_2} := \langle g, g \rangle_{BL_2}$. In contrast to the TV seminorm introduced earlier, the BL_2 seminorm is defined in the continuous domain and measures smoothness via the second-order derivatives. Smooth functions have a low BL_2 norm and vice-versa. Our minimization problem in this setting can now be formulated as

$$\min_f \quad \lambda \|f\|^2_{BL_2} + \sum_{m=1}^{M} (f(\mathbf{k}_m) - y_m)^2, \tag{3.17}$$

where the minimization is to be carried out over all functions f that are of the form (3.15). The first term in the above equation measures the smoothness of the solution $f(\mathbf{t})$ by the energy present in all of its second derivatives, and the second term is a fidelity term that attempts to fit the function $f(\mathbf{t})$ to the available data.

In order to solve this minimization problem, we need to choose a kernel function $\varphi(\mathbf{t})$. There are many choices available such as the bilinear or bicubic B-splines etc. In order to ensure consistency with the other recovery methods, and to obtain good quality approximations, we choose the optimal interpolating cubic B-spline proposed by Blu *et al.* [BTU01]. It is defined as

$$\beta^3_I(t) := \beta^3(t) - \frac{1}{6} \frac{d^2}{dt^2} \beta^3(t), \tag{3.18}$$

where $\beta^3(t)$ denotes the univariate uniform centered cubic B-spline. The corresponding bivariate kernel is obtained via a tensor product, i.e. $\varphi(t_1, t_2) = \beta_I^3(t_1)\beta_I^3(t_2)$. Observe that, with this choice of φ, the coefficient vector $\mathbf{c}$ is the same as the vector $\mathbf{x}$ (since the kernel is interpolating), and our minimization problem can be written equivalently (see [XAE12] for details) as

$$\min_{\mathbf{x}} \|\mathbf{S}\mathbf{x} - \mathbf{y}\|_2^2 + \lambda \mathbf{x}^T \mathbf{H} \mathbf{x}, \tag{3.19}$$

where the $N \times N$ matrix $\mathbf{H}$ is defined as

$$\mathbf{H}_{p,q} = \langle \varphi(\cdot - \mathbf{j}_p), \varphi(\cdot - \mathbf{j}_q) \rangle_{BL_2}. \tag{3.20}$$

Since (3.19) involves an ℓ_2 norm, we can differentiate with respect to $\mathbf{x}$ to obtain the following least-squares problem

$$(\mathbf{S}^T\mathbf{S} + \lambda \mathbf{H})\mathbf{x} = \mathbf{S}^T\mathbf{y}, \tag{3.21}$$

whose solution yields the minimizer of (3.19). This least-squares problem can be efficiently solved using the conjugate gradient method since the matrix $(\mathbf{S}^T\mathbf{S} + \lambda\mathbf{H})$ is symmetric and positive definite. The matrix $\mathbf{H}$ does not need to be explicitly computed or stored since its action on a vector $\mathbf{x}$ is equivalent to a filtering operation; the filter weights are obtained via (3.20).

Chapter 4

Results and Discussion

We generated volume rendered test images from datasets using our own volume renderer as well as Paraview [Squ07]. For recovery via dictionary learning, the datasets we used are different from those for all other methods in terms of resolution. We will describe them separately in the following sections.

4.1 Experimental Setup for Dictionary Learning

Inspired by various image processing applications of overcomplete dictionaries, we treat a volume rendered image as a collection of square pixel blocks. All results reported in this thesis utilize a block size of 16×16. We first train a dictionary for grey level images and based on the insight gained from the experiments, move on to treat color images.

4.1.1 Grey Level Images

When each color channel is treated independently, a separate dictionary needs to be trained to represent each channel. When using a prescribed dictionary, each channel has an independent sparse coding step. Therefore, it suffices to consider one channel only.

Training Data: We started by generating training images for the dictionary learning phase. This was done by rendering a chest CT dataset from 36 different viewpoints around the dataset; 35 of these were used for training and one was reserved for testing purposes. The renderings were obtained using the batch processing capabilities of ParaView [Squ07]. The images, each having a resolution of 288×512, were rendered in color using a transfer function with three dominant colors; some of the views are shown in Fig. 4.1. These images were then converted to greyscale while the original color images were saved for later use. In

order to reduce the amount of training data and to simplify the subsequent training phase, we chose camera parameters so that the dataset was tightly centered within the camera view with few background pixel blocks.

Dictionary learning: We used the K-SVD Matlab toolbox that accompanies Elad's book [Ela10] to train a dictionary based on the grayscale images obtained previously. The training blocks were obtained by sliding a square 16×16 window over all the training images without any overlap. The K-SVD training algorithm supports different error thresholds ϵ and sparsity levels L when applying OMP. After some initial experimentation and based on previously published results [MAB06], we chose a dictionary size of 256×1024 and trained the dictionary with a maximum target sparsity of $L = 10$. The training process took 7.1 minutes on a single thread implementation running on an Intel®Core$^{\text{TM}}$i7-4770K @ 3.50GHz CPU with 16 GB RAM.

Prescribed DCT dictionary: In order to compare the quality of the trained dictionary with a prescribed dictionary, we generated an overcomplete DCT dictionary of the same size. A DCT dictionary matrix C of size $M \times N$ corresponding to 1D atoms of size M is obtained according to:

$$C_{i,j} = \cos\big(\frac{\pi}{N}(i-1)(j-1)\big), \quad i = 1, \ldots, M, \quad j = 1, \ldots N. \tag{4.1}$$

A DCT dictionary of size $M^2 \times N^2$ corresponding to $M \times M$ 2D blocks is then obtained by taking the kronecker product of C with itself. Before using the 2D DCT dictionary, we normalized the columns to have zero mean and a unit 2-norm.

Missing Pixels: If an image has some of its pixels missing, it is possible to use OMP to recover a sparse approximation that is most consistent with a given dictionary. Let us assume that a singal $\mathbf{y}$ has the representation $\mathbf{y} = \mathbf{D}\mathbf{x}$, where $\mathbf{x}$ is a sparse coefficient vector. Let $\tilde{\mathbf{y}}$ be a subsequence of $\mathbf{y}$ obtained by removing some of the components, and let $\mathbf{S}$ be the matrix that accomplishes the subsampling process, i.e. $\mathbf{S}\mathbf{y} = \tilde{\mathbf{y}}$. The matrix $\mathbf{S}$ has as many

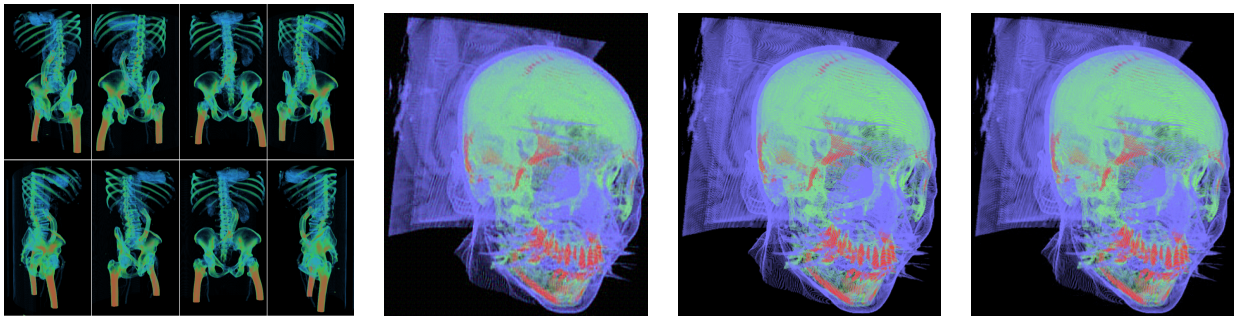
rows as the number of entries in $\tilde{\mathbf{y}}$ and as many columns as the number of entries in the original signal $\mathbf{y}$. Each row of $\mathbf{S}$ has a unit entry corresponding to the index of the entry in $\mathbf{y}$ that is to be retained, and has zeros everywhere else. The subsampled representation can therefore be written as $\mathbf{SDx} = \mathbf{Sy}$, or equivalently, $\tilde{\mathbf{D}}\mathbf{x} = \tilde{\mathbf{y}}$ where $\tilde{\mathbf{D}} := \mathbf{SD}$ is a decimated dictionary. We can now solve the modified problem

$$\min_{\mathbf{x}} \|\mathbf{x}\|_0 \, s.t. \tilde{\mathbf{y}} = \tilde{\mathbf{D}}\mathbf{x} \tag{4.2}$$

via OMP to recover a sparse coefficient vector $\mathbf{x}$ that is most consistent with the subsequence $\tilde{\mathbf{y}}$ in the decimated dictionary $\tilde{\mathbf{D}}$. Computing $\mathbf{Dx}$ then recovers an approximation of the original signal $\mathbf{y}$.

4.1.2 RGB Images

There are several options available when considering dictionary learning in the context of RGB images. A straightforward option is to train three dictionaries for each channel separately. This leads to a three-fold increase in the encoding time as each channel needs to be encoded separately. Another option is to convert the RGB images to an indexed representation and apply dictionary learning to the indexed images. This approach however can lead to serious artifacts as it is akin to pre-classification in volume rendering. Yet another approach — and the approach we prefer — is to train the dictionary based on 3D RGB blocks ($16 \times 16 \times 3$ in our case). Color channels in volume rendered images are not completely independent and can be strongly correlated. This approach has the advantage that it can detect and learn the proper relations between the channels [MES08]. It leads to a bigger overcomplete dictionary since the number of rows experiences a three-fold increase; the number of dictionary atoms needs to be increased as well in order to obtain a dictionary that has a sufficient degree of overcompleteness. On the other hand, the increase in dictionary size can lead to possibly sparser representations owing to the learned correlations between the channels.

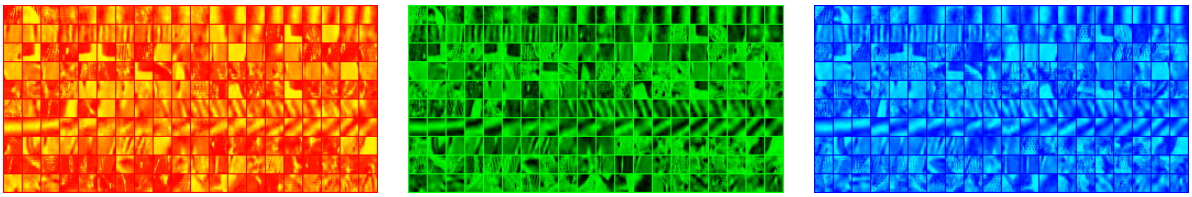


(a) Training images (b) Trained (c) Original (d) DCT

Figure 4.1: Dictionary learning involves the training of column vectors of a dictionary matrix from a set of exemplar vectors so that the exemplar vectors can be represented via a sparse linear combination of the trained vectors. A color dictionary was trained by using $16 \times 16 \times 3$ pixel blocks extracted from renditions of a chest dataset captured from 35 different viewpoints (a). Given a test image obtained from a different dataset using different rendering parameters (c), the trained dictionary can be used to approximate the test image with only 30 coefficients (b). The sparse encoding procedure finds the right combination of coefficients needed to correctly blend the trained vectors, and also leads to a dissipation of ringing artifacts. In contrast, an approximation obtained via a preconstructed DCT dictionary using 30 coefficients per channel (90 altogether) exhibits stronger ringing artifacts (d).

Training and Comparison: As mentioned earlier, we generated color images using a transfer function that had dominant red, green and blue hues (see Fig. 4.1). This ensures that the dictionary has enough color variation in the three channels so that it can be used to represent color images that are very different from the training images. The K-SVD algorithm was then used to train a 768×2304 dictionary. For training and reconstruction purposes, the $16 \times 16 \times 3$ blocks were linearized into a 768×1 column vector whose first 256 entries correspond to the red channel, the next 256 correspond to green, and the remaining 256 to blue. Some of the trained blocks are visualized in Fig. 4.2. The training process took 47.3 minutes.

We compared the trained dictionary with a component-wise DCT dictionary having the same size as that used in our grayscale tests (256×1024 for each channel). By design, our comparison favors the DCT dictionary as it has a greater column-to-row ratio and therefore has greater redundancy. A number of scenarios were then compared as explained in Section 4.6.



(a) Red channel (b) Green channel (c) Blue channel

Figure 4.2: The first two hundred blocks from the trained dictionary - each channel is visualized separately by extracting the red, green and blue portions from the dictionary atoms. Since the dictionary can have negative entries, each channel was mapped to the range $[0, 1]$ for display purposes. Observe that the red and blue channels are more saturated as compared to the green channel.

4.2 Experimental Setup for Other Methods

The focus of recovery via dictionary learning lies on the level of sparsity or compression rate while our other methods intend to deal with higher percentage of missing pixels and in the meantime produce high quality recovery results. For all other methods, test images were generated at a resolution of 1200×1200 on a workstation with a quad-core, 3.4 Ghz Intel®Core$^{\text{TM}}$i7-3770 CPU with 16GB RAM. The data we used has a resolution of around $256 \times 256 \times 256$ and with a resolution of 1200×1200 for our output images we could have enough pixels per voxel to guarantee the smoothness of the image. At higher resolution, the high-quality tricubic interpolation can produce smoother image while have slightly better quality than images with lower resolution (600×600). We did not go beyond the resolution of 1200×1200 because higher ones are not very different from the one we generated while the rendering time is longer. All of our recovery experiments were conducted in Matlab where we used the NESTA solver [BBC11] to carry out ℓ_1 (CS-Wavelet, CS-Gradient) and TV minimization. We used Matlab's conjugate gradient solver to solve the least-squares minimization (SS) problem (3.19). For parameter settings, we used 20 for the standard deviation of the blurring filter Φ for the CS-Wavelet approach which, according to the results of Sen and Darabi [SD11], balances the tradeoff between incoherence and blurring effects. We used the same value of 10^{-2} for the tolerance and stopping criteria of the NESTA solver

in our CS-Wavelet, CS-Gradient and TV experiments. This value was empirically chosen to provide a good tradeoff between recovery quality and runtime. For the least-squares solver (SS), we used the value 10^{-2} for the regularization parameter λ as suggested by Xu *et. al* [XAE12].

We recovered the images from a fraction of the pixels. We experimented with different percentages of pixels that are removed via two different pixel distribution algorithms explained in the following section. To measure recovery quality, we computed the peek signal-to-noise ratio (PSNR) values measured in decibels (dB). The PSNR is a well-known quality metric and is a good way to quantify large differences in recovery trends exhibited by the different techniques. However, the PSNR is not a perceptual measurement scheme. In order to measure the human perceptual quality, we utilized two more measurement systems. We computed error images in the CIELUV colorspace according to the work of Ljung *et al.* [LLYM04]. Moreover, we computed and generated Structural SIMilarity (SSIM [WBSS04]) index as well as the SSIM maps to measure the recovery quality. The Structural SIMilarity (SSIM) index is a method for measuring the similarity between two images. The SSIM index can be viewed as a quality measure of one of the images being compared, provided the other image is regarded as of perfect quality [WBSS04]. In addition to measuring the recovery quality, we also measured the performance of each method with respect to the timing for recovery.

4.3 Choice of Distribution Algorithm

Choosing a pixel distribution algorithm wisely is of significant importance as our goal is to recover volume rendered images from a small fraction of the pixels. A straightforward way of choosing pixels is the random distribution, obtained by randomly drawing a number between 1 and N where N is the total number of pixels. This strategy however leads to inhomogeneous regions (Fig. 4.3). A better strategy is to distribute the given pixels as uniformly as possible so that the overall discrepancy [PH10] is low. A potential distribution that achieves low discrepancy (LD) can be obtained by the Poisson disk sampling algorithm [PH10]. This algorithm achieves very good blue-noise distributions. However, a disadvantage is that it is not progressive, i.e. a distribution with a high percentage of coverage does not contain a distribution with a low percentage of coverage. This is an important property for compressive rendering as it allows for the progressive update of an image. A distribution that does satisfy this property is provided by Anderson's pixel shuffle algorithm [And93]. This algorithm is based on the Fibonacci numbers and progressively adds (or removes) pixels so that the resulting set of pixels maintains a good discrepancy (Fig. 4.3). Fig. 4.4 demonstrates the progressive property of this pixel shuffle algorithm [And93].

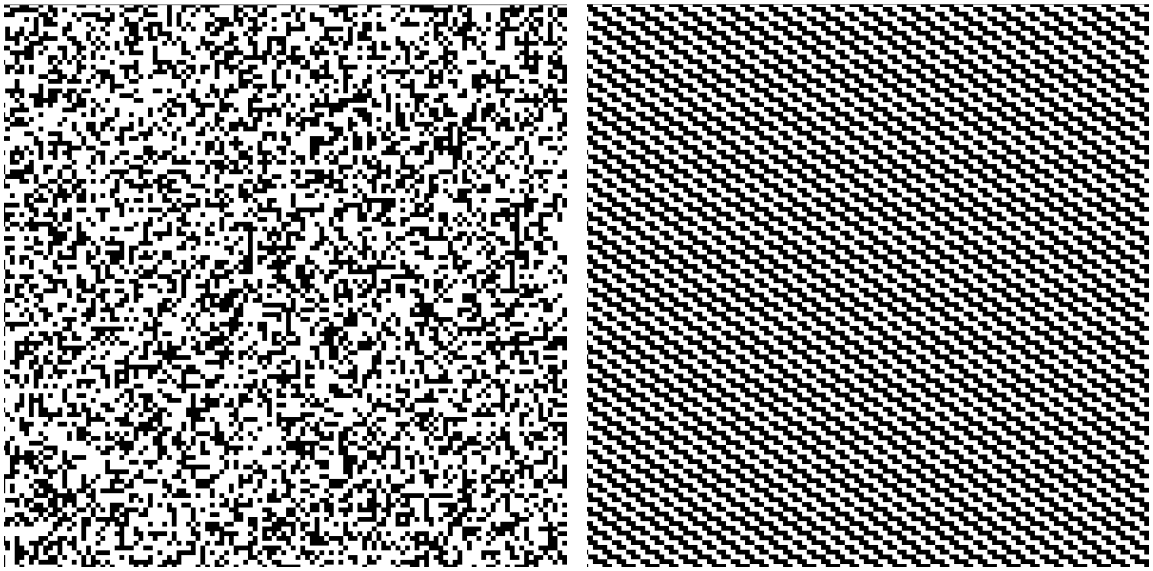


Figure 4.3: Masks with 50% missing pixels; left: random, and right: LD via pixel shuffle.

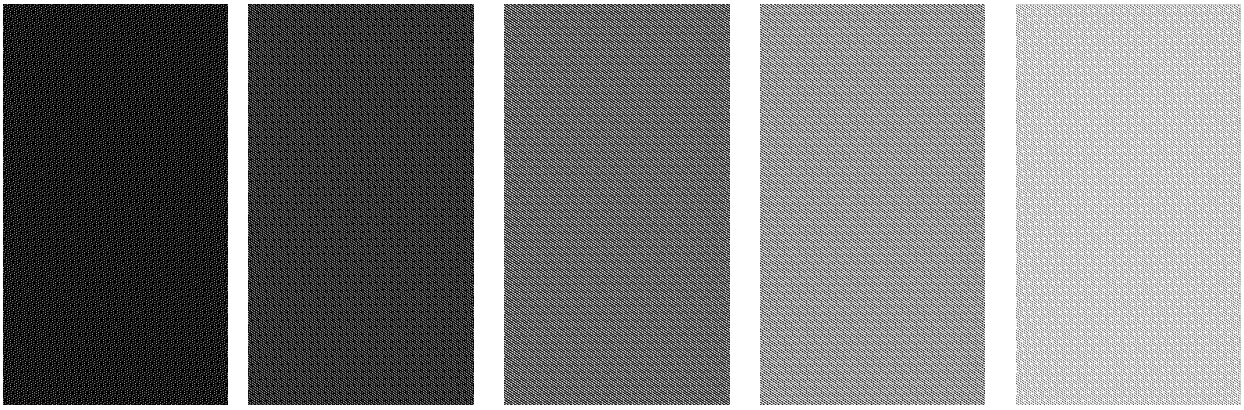


Figure 4.4: Distribution of pixels generated by the pixel shuffle algorithm. White pixels are on and black pixels are off.

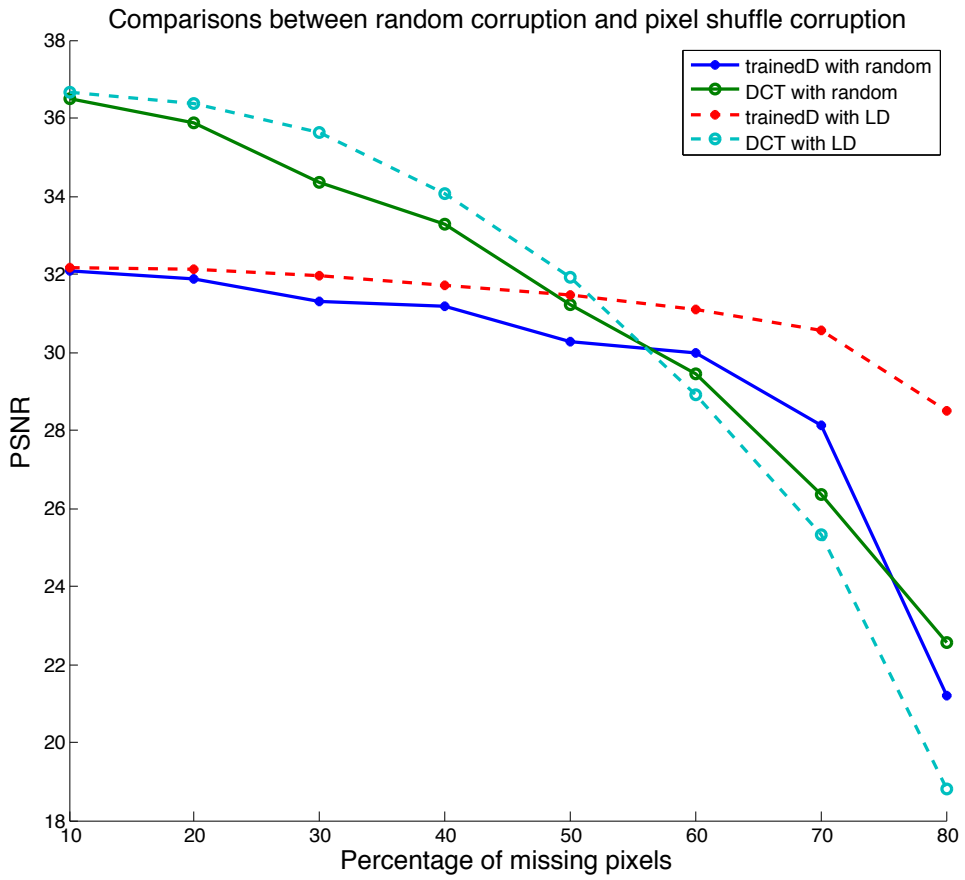
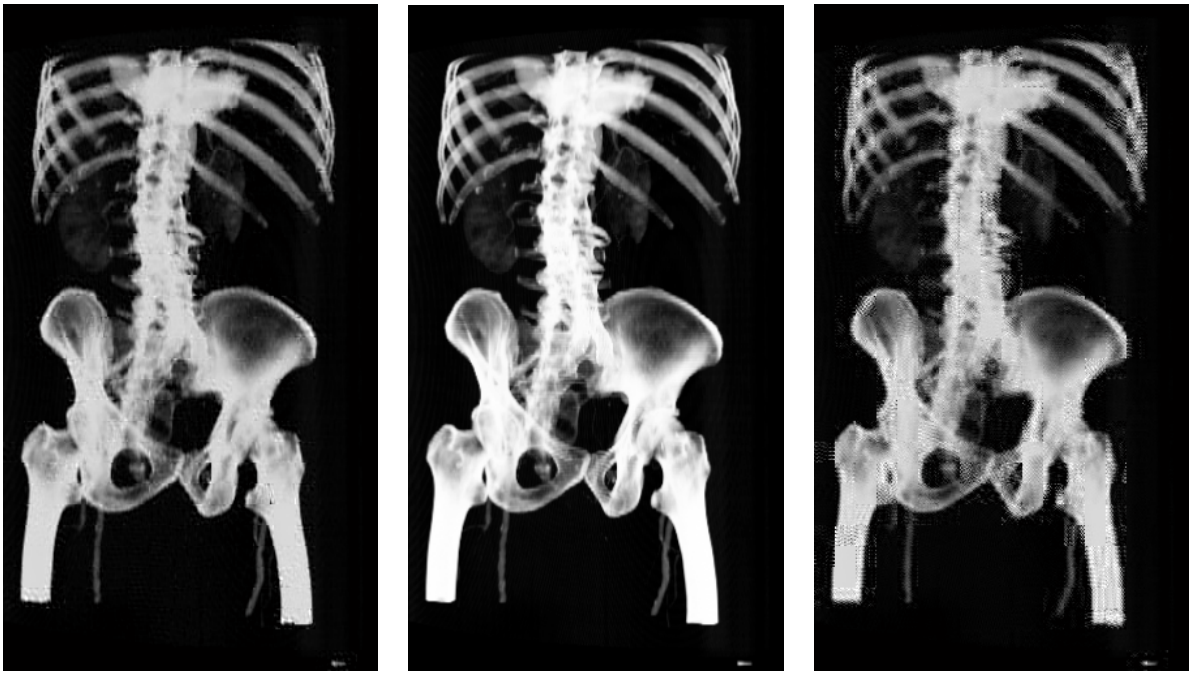


Figure 4.5: Comparison between different distribution algorithms

4.4 Missing Pixel Comparisons for Dictionary Learning

Random Distribution vs. Pixel Shuffle As discussed in Section 4.3, choosing a pixel distribution algorithm wisely is of significant importance to the recovery results. We ran experiments with the trained dictionary and the DCT dictionary in conjunction with different distribution algorithms. Fig. 4.5 represents the differences between recovery results with these two algorithms. We can see that the trained dictionary with low discrepancy (LD) can produce slightly better recovery results for all different percentages of missing pixels. Thus, in the following sections, unless otherwise motioned, we apply low discrepancy distribution to run experiments with our trained dictionary as well as DCT dictionary.



(a) Trained, PSNR:31.2 (b) Original image (c) DCT, PSNR:22.97

Figure 4.6: Reconstruction results with 70% missing pixels.

4.4.1 Grey-level Images:

In order to compare the quality of the two dictionaries, we encoded the original test image as well as incomplete versions obtained by discarding a given percentage of the pixels. Fig. 4.6 shows the original image recovered from only 30% of the pixels using the two dictionaries whereas Fig. 4.7 compares the peek signal-to-noise ratio (PSNR) for varying fractions of the number of missing pixels. As expected, the trained dictionary outperforms the DCT dictionary for this test case. Since the pixel blocks in the test image are very similar to the training blocks, the trained dictionary is able to recover an image with an acceptable quality with as few as 30% of the pixels. On the other hand, the DCT dictionary performs well in the presence of complete information but the PSNR quickly deteriorates as the number of missing pixels increases.

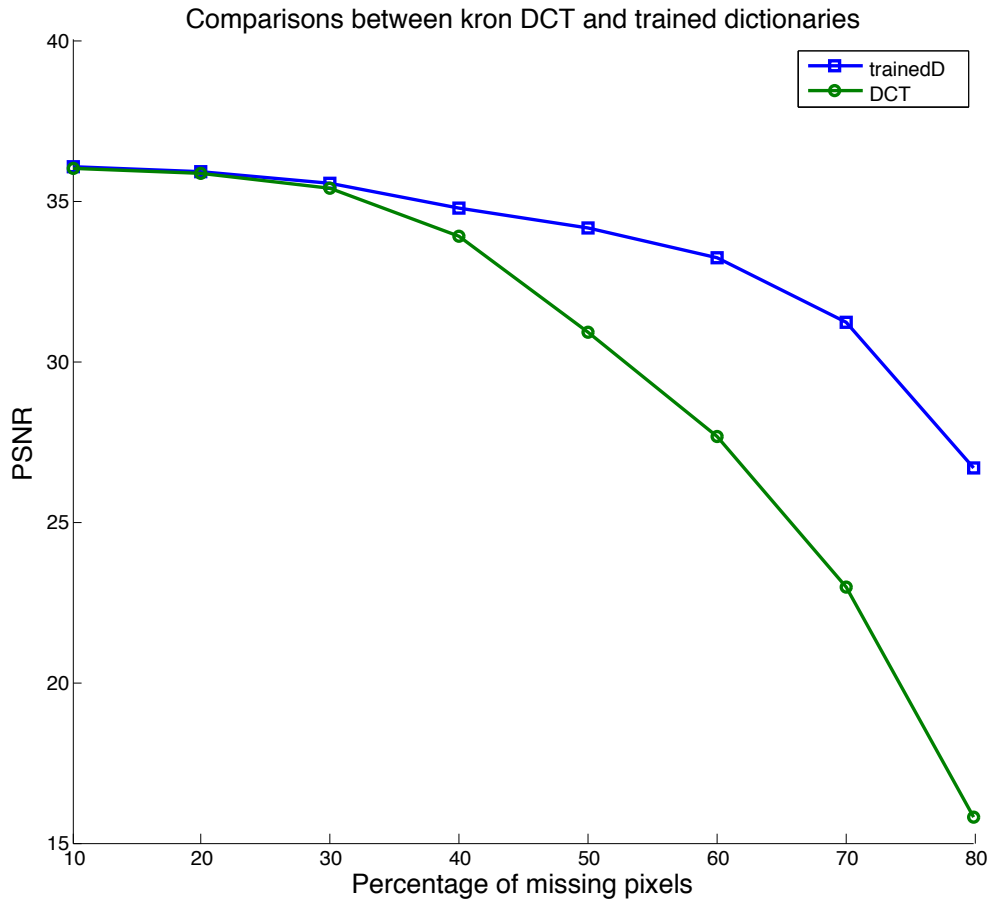


Figure 4.7: Comparison between the trained dictionary (grey level) and the DCT dictionary. For both dictionaries, each image was encoded with a sparsity of $L = 30$.

4.4.2 RGB Images

We tested the qualitative and quantitative performance of the two dictionaries when recovering the test image with varying fractions of the original pixels. We also tested different values of the sparsity level L in the encoding phase. For all the tests, we also recorded the total encoding time and the total decoding time.

Fig. 4.8 plots PSNR values as a function of the total percentage of missing pixels for two encoding scenarios: $L = 30$ and $L = 60$. It should be stressed that for the DCT dictionaries, the parameter L refers to the sparsity of each channel, i.e. when $L = 30$, there can be upto 30 non-zero coefficients per channel. On the other hand, for the trained dictionary, the parameter L corresponds to the actual sparsity as all three channels are encoded in a

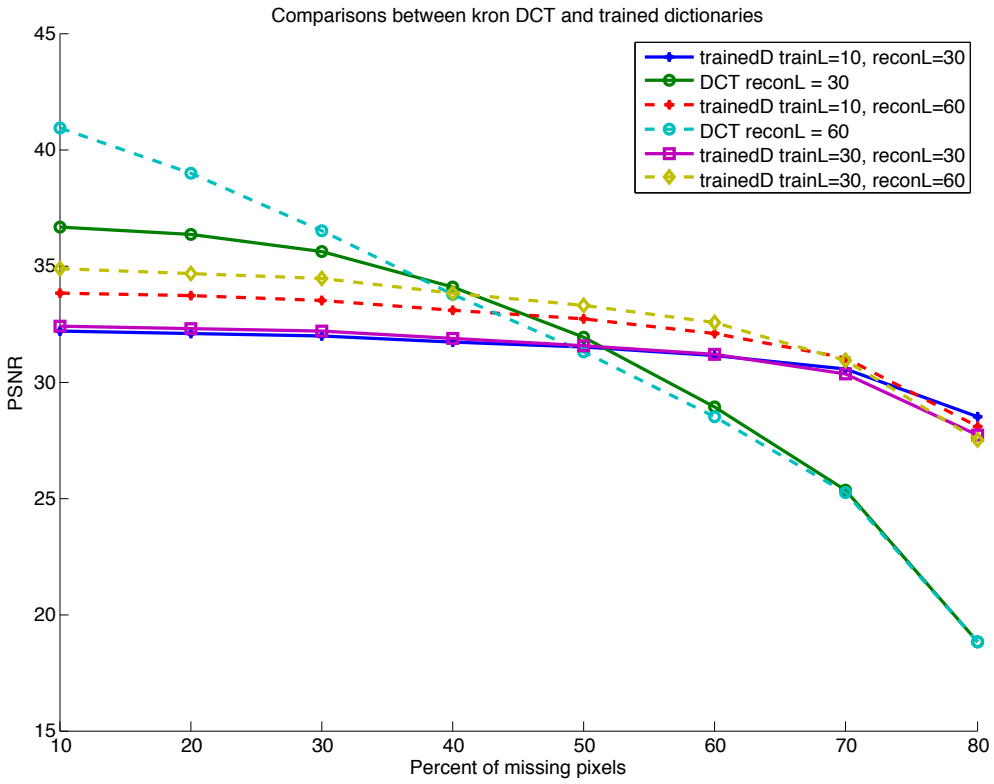
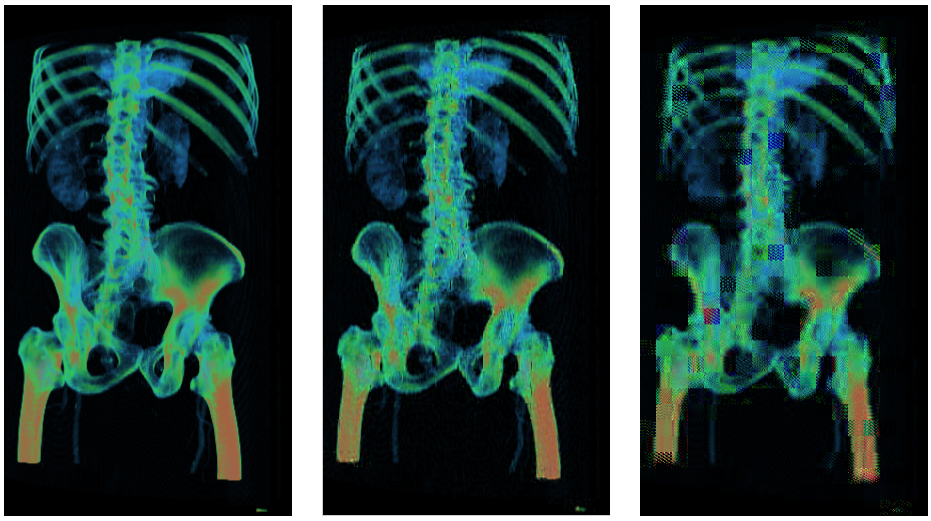
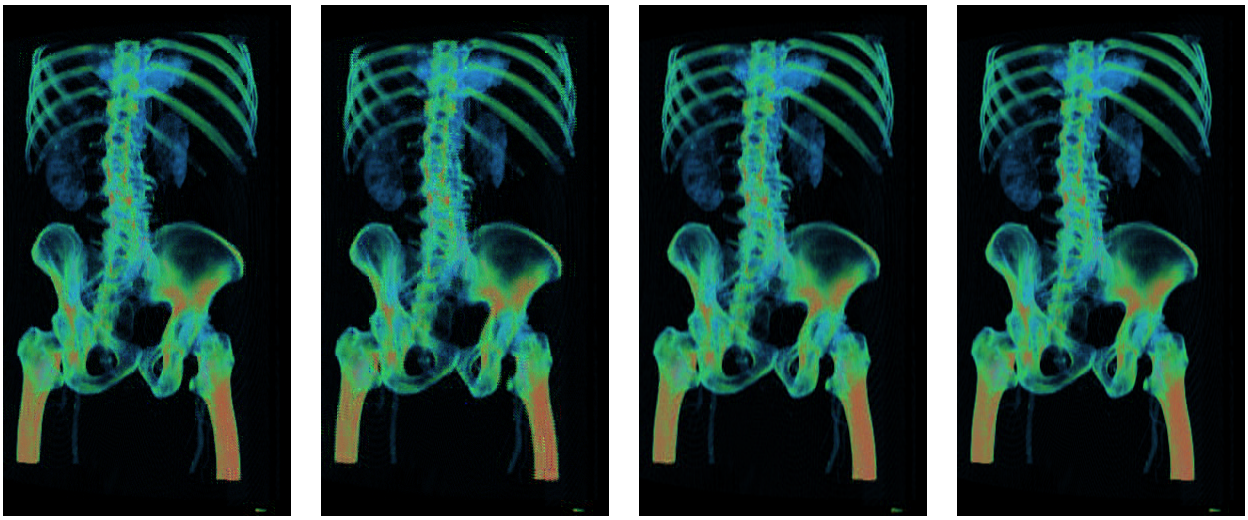


Figure 4.8: Quantitative comparison of trained (RGB) and DCT dictionaries. Horizontal axis represents the total fraction of missing pixels and the vertical axis represents the PSNR values of the recovered test image.



(a) Original image (b) 80% Trained (c) 80% DCT



(d) 60% Trained (e) 60% DCT (f) 10% Trained (g) 10% DCT

Figure 4.9: Recovering the test image from a fraction of the pixels ($L = 30$). For quantitative comparisons and timing results, please see Table 4.1.

combined format.

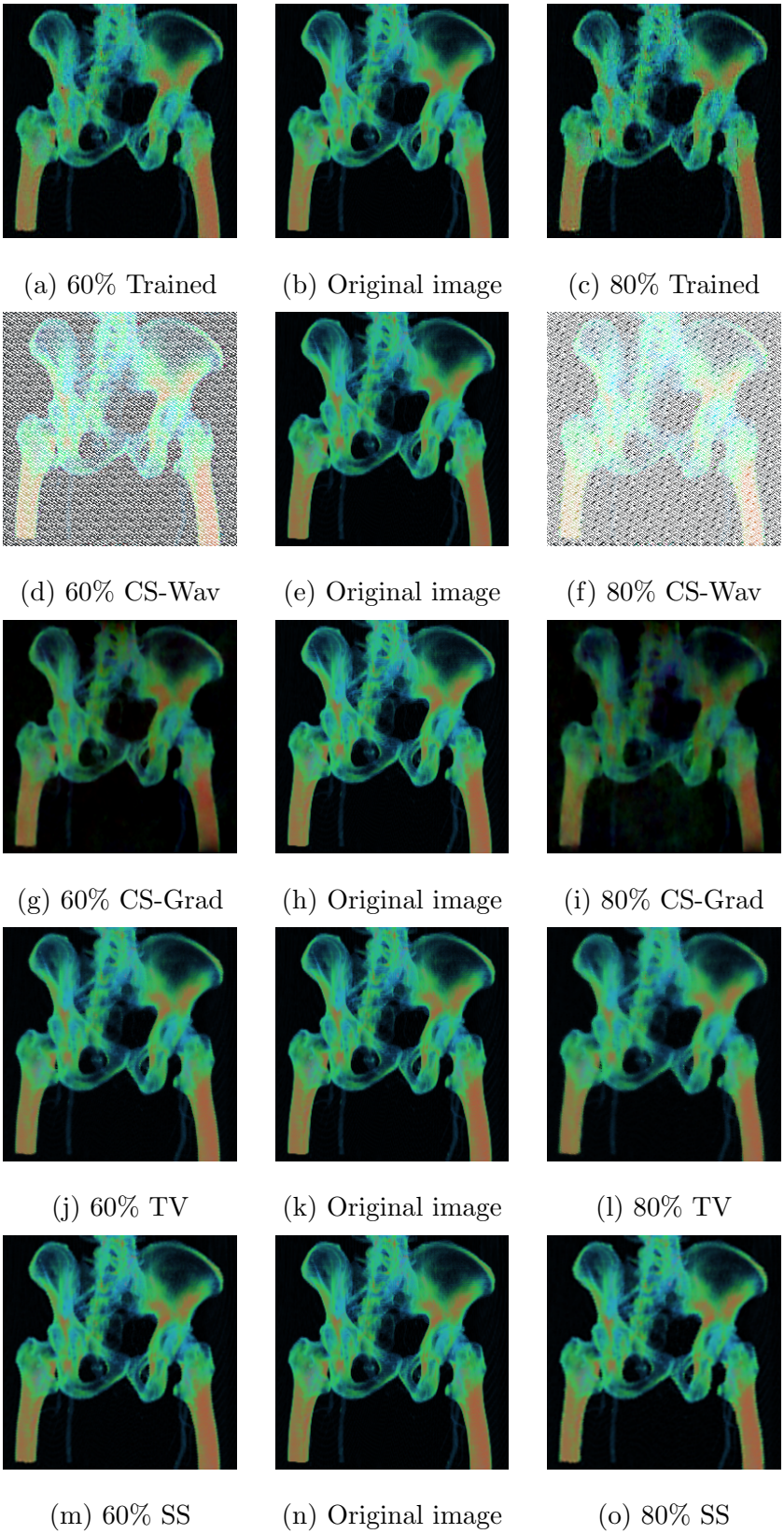
It is immediately obvious that the trained dictionaries have a much gradual fall off as compared to the DCT dictionary. The DCT dictionary on the other hand does better in the presence of complete information. This is to be expected since the DCT encoded representations have three times as many non-zero coefficients as their trained counterparts.

When the dictionary is trained with a sparsity of $L = 10$, and the images are encoded with a sparsity of $L = 30$, the trained dictionary is much better in the presence of incomplete information as compared to the DCT dictionary; the point where the two curves cross is just around the 50% mark. If we allow more non-zero coefficients in the encoding phase ($L = 60$), the crossing point shifts to the left and is closer to the 40% mark. Also notice that the quality of the recovery is more sensitive to L in the encoding phase as compared to the training phase. Unless otherwise stated, all subsequent results are with respect to the dictionary trained with $L = 10$.

To compare the visual quality of the reconstructions, we picked three sets of images with different percentages of missing pixels: 80%, 60% and 10%. The corresponding images are shown in Fig. 4.9 and the timing results are shown in Table 4.1 in Section 4.7. As expected, reconstructed images with trained dictionary are visually better than the DCT ones in both 80% and 60% missing cases. At 80%, the DCT image exhibits strong block artifacts which are substantially reduced in the trained case. Images reconstructed with 10% missing pixels look pretty similar visually even though there is about a 6dB difference between them.

4.5 Missing Pixel Comparisons For All Methods

In this section, we compare and contrast the recovery results for all methods. Following the discussions on recovery via dictionary learning, we first compare the qualitative (Fig. 4.10) and quantitative (Fig. 4.11) results between dictionary learning and other methods on the data used for training the dictionary. Next, we empirically compare other methods on other rendered images, such as Physically-based rendered images, iso-surface rendered images as well as some synthetic images.



(a) 60% Trained (b) Original image (c) 80% Trained

(d) 60% CS-Wav (e) Original image (f) 80% CS-Wav

(g) 60% CS-Grad (h) Original image (i) 80% CS-Grad

(j) 60% TV (k) Original image (l) 80% TV

(m) 60% SS (n) Original image (o) 80% SS

Figure 4.10: Comparisons between all methods

4.5.1 Comparisons Between Dictionary Learning And Other Methods

Fig. 4.10 illustrates the qualitative comparisons between all methods on the data used to train the dictionary while Fig. 4.11 demonstrates the quantitative differences between all methods. We can reckon that CS-Wavelet cannot produce acceptable results for images with more than 50% missing pixels. CS-Gradient, together with random distribution, can produce slightly better results with color shifting artifacts. Moreover, recovery via trained dictionary can produce acceptable results for images with fewer than 60% missing pixels. In the 80% case, blocky artifacts have been introduced all over the image. Notably, our TV and SS methods can produce consistent recovery results even with 80% missing pixels. Thus, TV and SS outperform all other methods in terms of quality of the recovery and more discussions about performance are presented in Section 4.7.

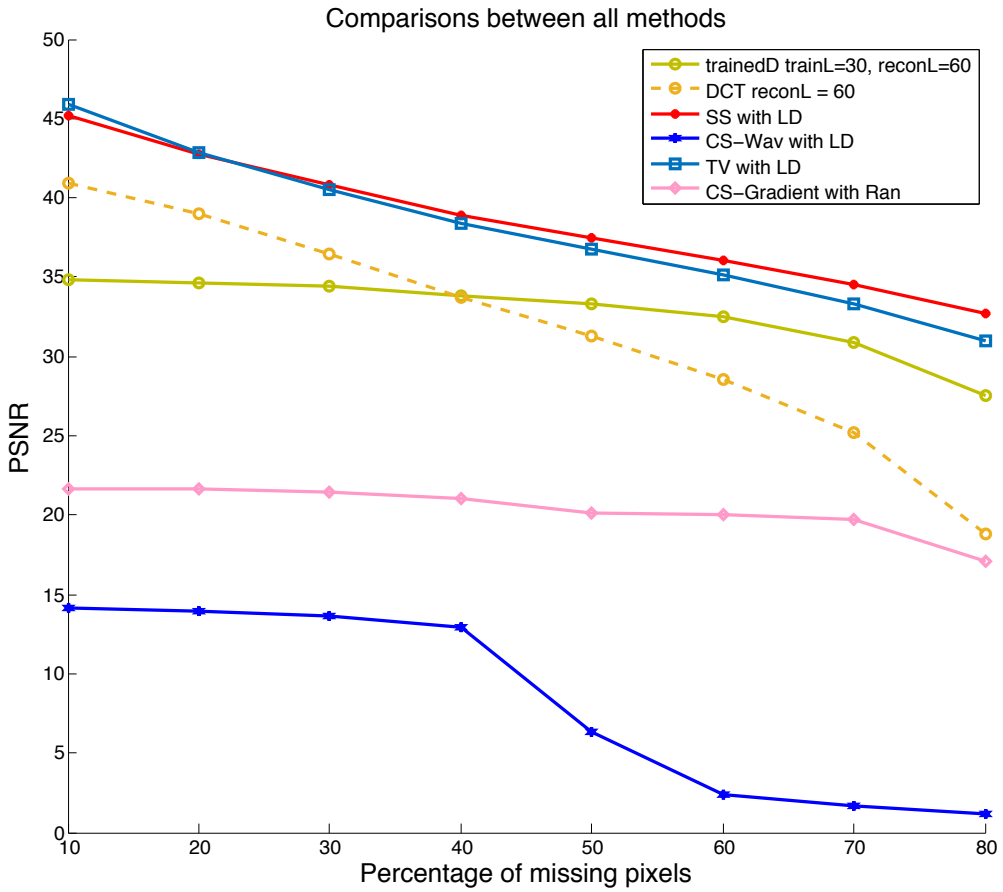


Figure 4.11: Comparison between dictionary learning and other methods

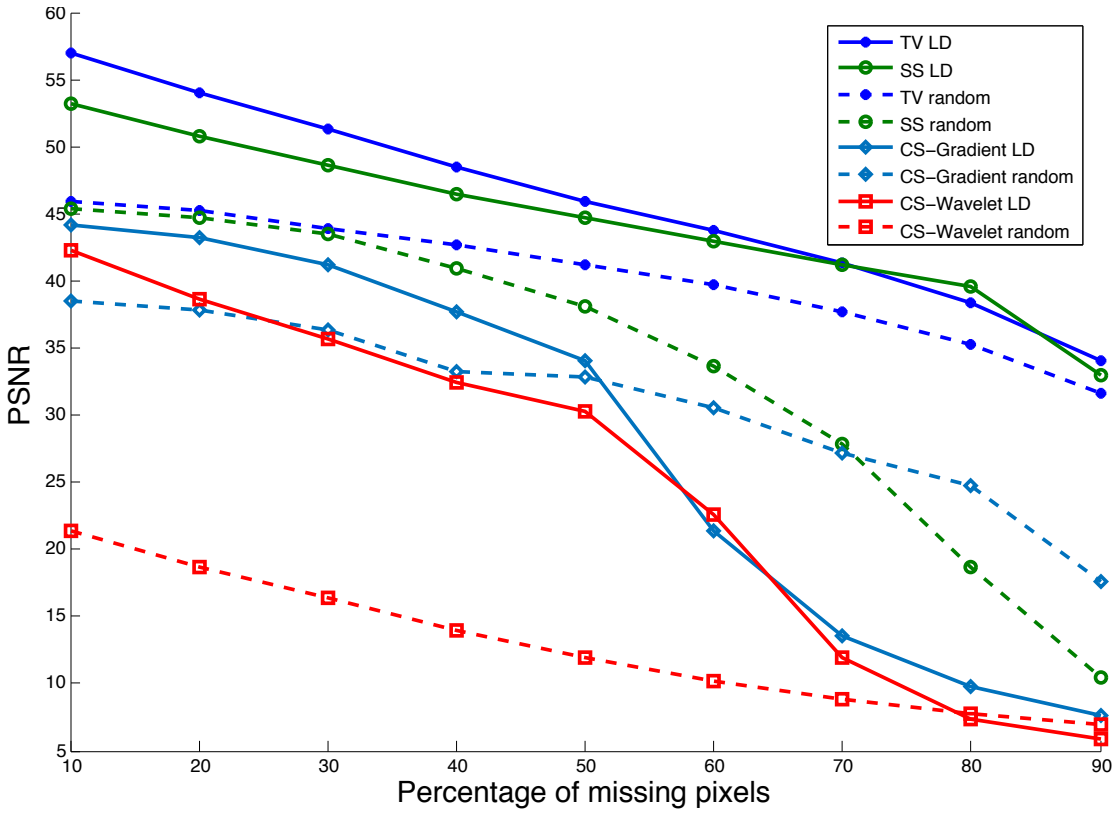


Figure 4.12: Random vs. LD distributions.

4.5.2 Comparisons Of Other Methods

In addition to the previous data, we also ran experiments on other datasets. All images used here are with resolution of 1200 × 1200 and we tested all methods except for recovery via dictionary learning due to the larger image size which leads to much longer training time (see Section 4.7 for details of performance comparisons).

Random Distribution vs. Pixel Shuffle We used the two aforementioned distribution algorithms to compare the recovery quality of all the methods. The quantitative results for the head dataset are shown in Fig. 4.12, and some of the qualitative results are shown in Fig. 4.13 and Fig. 4.14. We can observe that our CS-Gradient method produces slightly better results compared to the CS-Wavelet method.

The CS-Wavelet method seems to be highly sensitive to the distribution of pixels. In our tests, we observed that, when the percentage of missing pixels is high, the random distri-

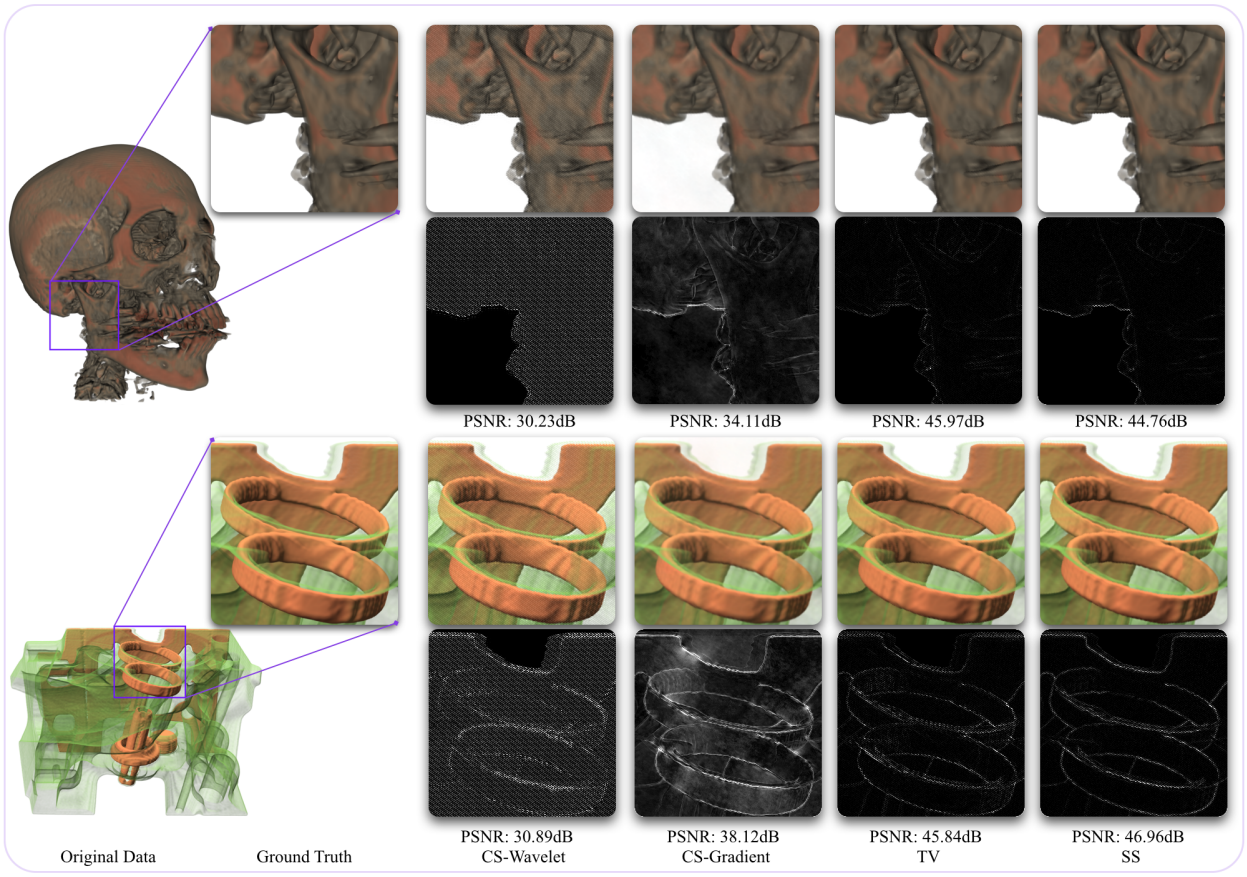


Figure 4.13: Recovery results for all methods with LD distribution (50% missing pixels). The grayscale images indicate the magnitude of the error computed in the CIELUV space; the error range range $[0, 20]$ is linearly mapped to a grayscale colormap.

bution leads to strong speckling artefacts. The LD distribution achieves a lower coherence (Fig. 3.1) and thefore yields better results. However, it also exhibits directional artefacts (Fig. 4.13: second column). In comparison, our CS-Gradient method fares much better (Fig. 4.12). It favours the random distribution when the fraction of missing pixels is high. This is to be expected as the random distribution leads to partial Fourier matrices with better incoherence when the fraction of missing pixels is high (Fig. 3.1). Additionally, we also did not notice any objectionable artefacts with this method (Fig. 4.13: third column). However, notice that both CS-Wavelet and CS-Gradient quickly deteriorate when the fraction of missing pixels is high.

The smoothness-based methods (TV and SS) significantly outperform the CS-based

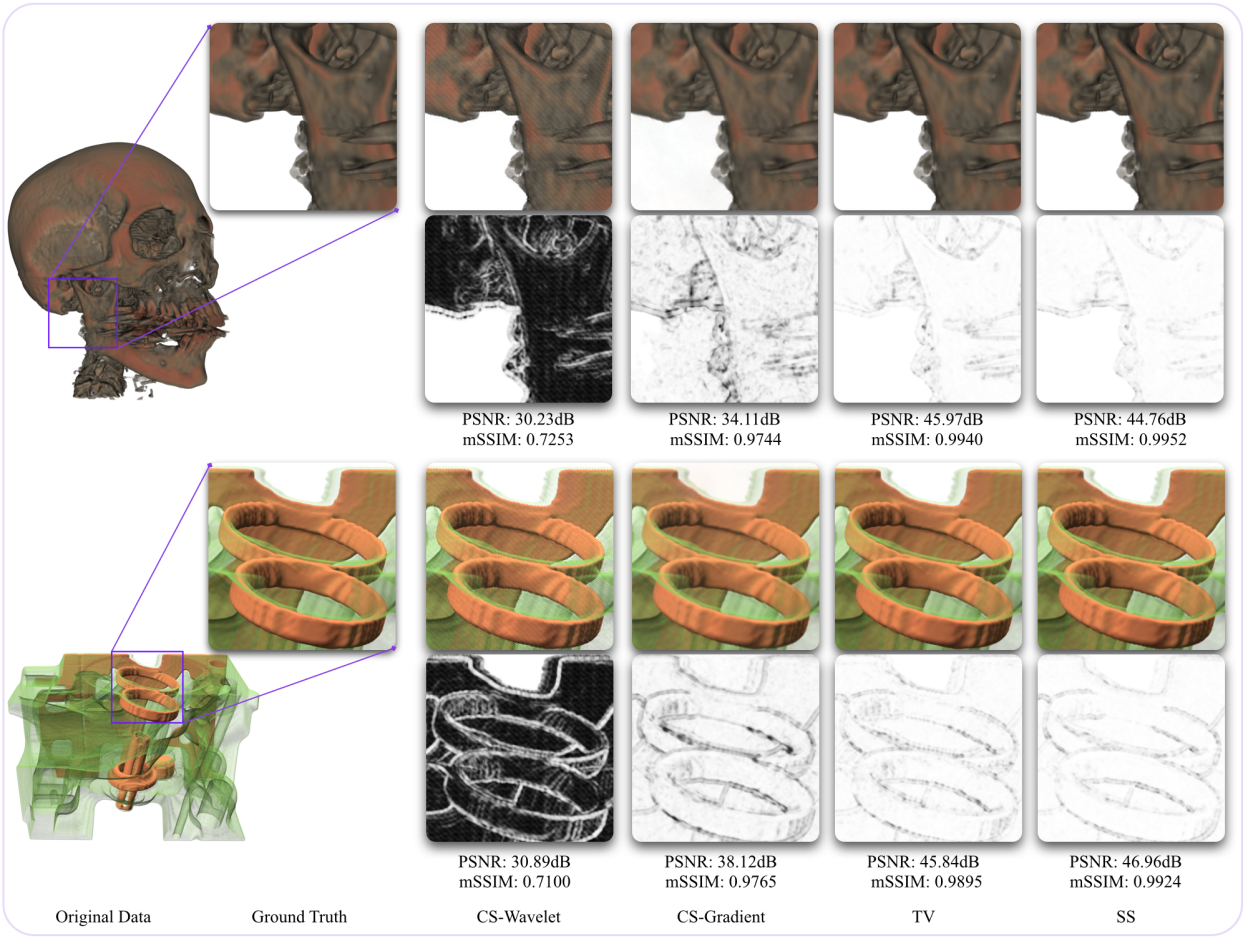


Figure 4.14: Recovery results for all methods with LD distribution (50% missing pixels). The mSSIM value is the mean SSIM index value between 2 images. The second and fourth rows are SSIM maps of the compressed images, where brightness indicates the magnitude of the local SSIM index (squared for visibility) [WBSS04]. It is similar to the error images shown in Fig. 4.13. However, their colormaps are inverted. Moreover, PSNR values are consistent with these perceptual measurements.

methods. Both seem to favour the LD distribution over the random distribution, the differences are indeed quite stark (Fig. 4.12). In terms of reconstruction quality, the two methods, in conjunction with the LD distribution are quite comparable when the fraction of missing pixels is high (Fig. 4.1 and Fig. 4.13). The TV method seems to have an edge over SS when the fraction of missing pixels is low. In the following results, we focus on the smoothness based methods and compare them with CS-Wavelet in combination with the LD distribution.

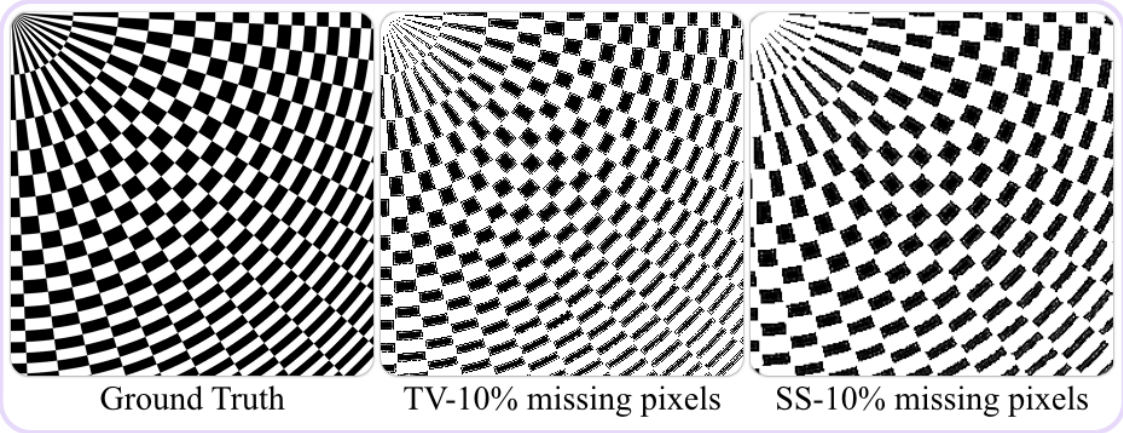


Figure 4.15: Recovery results for high frequency texture image.

Sensitivity to Image Content To compare the sensitivity of the algorithms to images with different smoothness characteristics, we used Paraview to render DVR images of the foot dataset and isosurface images of the aneurysm dataset. Some qualitative results are shown in Fig. 4.16. In terms of recovery quality, TV and SS produce similar results; SS recovery is somewhat sharper in the boundary regions while the TV method seems to preserve edges better. From Fig. 4.16, we can see that our TV and SS methods can produce consistent recovery results over different missing pixels. The higher PSNR values for the foot images corroborate the fact that these methods perform much better when the image content is slowly varying. The results for the aneurysm dataset show some degradation when the fraction of missing pixels is high. However, it is not as severe as it is for the CS-based methods.

Moreover, we tested our TV and SS methods on some volume rendered image with higher quality in Fig. 4.17. This crocodile mummy image is an example of volume ray casting and it was rendered by Fovia's High Definition Volume Rendering engine [Wik15b]. From Fig. 4.17, it is easy to reckon that our TV and SS methods can produce very good and consistent recovery even with 70% of missing pixels and it is as expected since the focus of our methods is on volume rendered images.

In order to investigate the importance of the smoothness of the image, we experimented

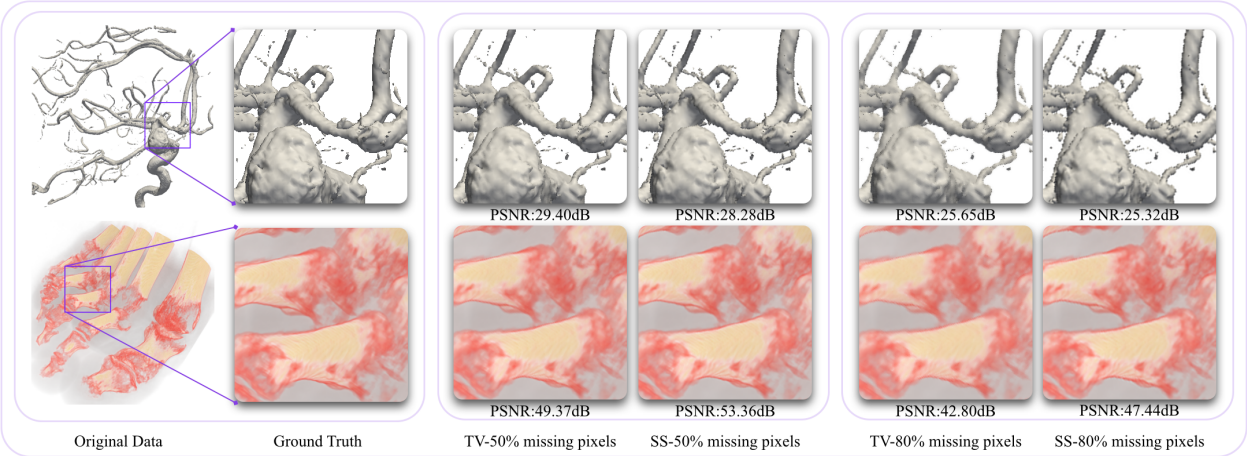


Figure 4.16: Recovery results for foot and aneurysm data via TV and SS

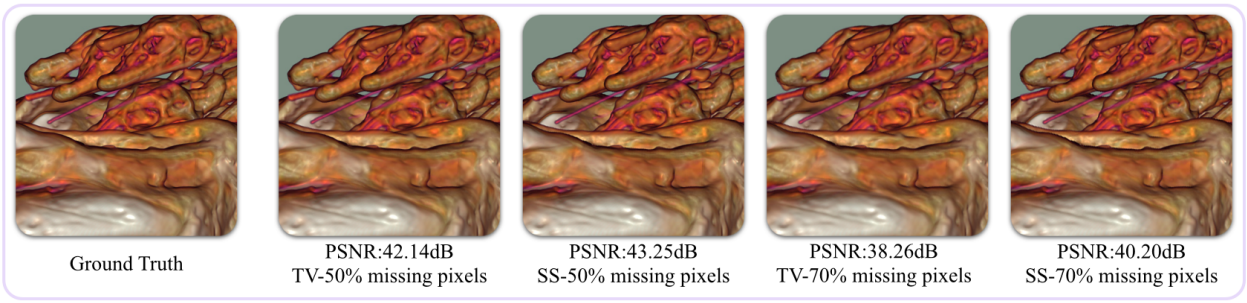


Figure 4.17: Recovery results for a crocodile mummy with juveniles on its back [Wik15b] via TV and SS

with a synthetic texture image that has both high and low frequencies alike (Fig. 4.15). Both TV and SS methods cannot produce acceptable recovery even with 10% missing pixels. As summarized in Table 4.7, this image lacks smoothness which is a key assumption made by all the recovery algorithms.

Other Applications The methods presented in this thesis can also be applied to ray-traced images. We tested the recovery results on the watch image generated using the physically-based renderer LuxRender. The original image took several hours to render. Fig. 4.18 shows the comparisons between different algorithms. As expected, our TV and SS methods can produce good recovery results even with 50% missing pixels. For 70% missing pixels, the results are acceptable but do exhibit subtle artefacts.

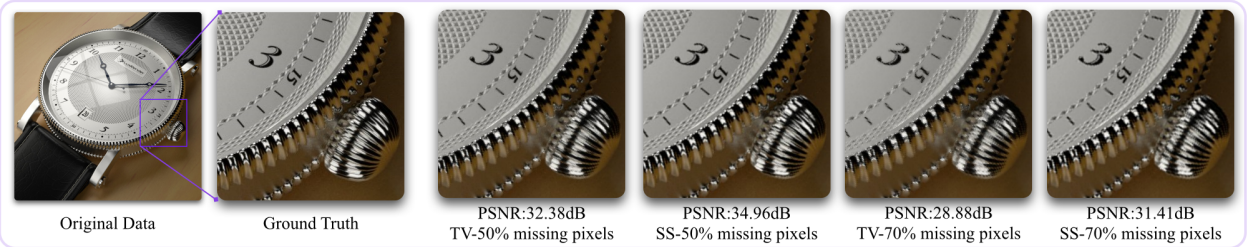


Figure 4.18: Recovery results via TV and SS

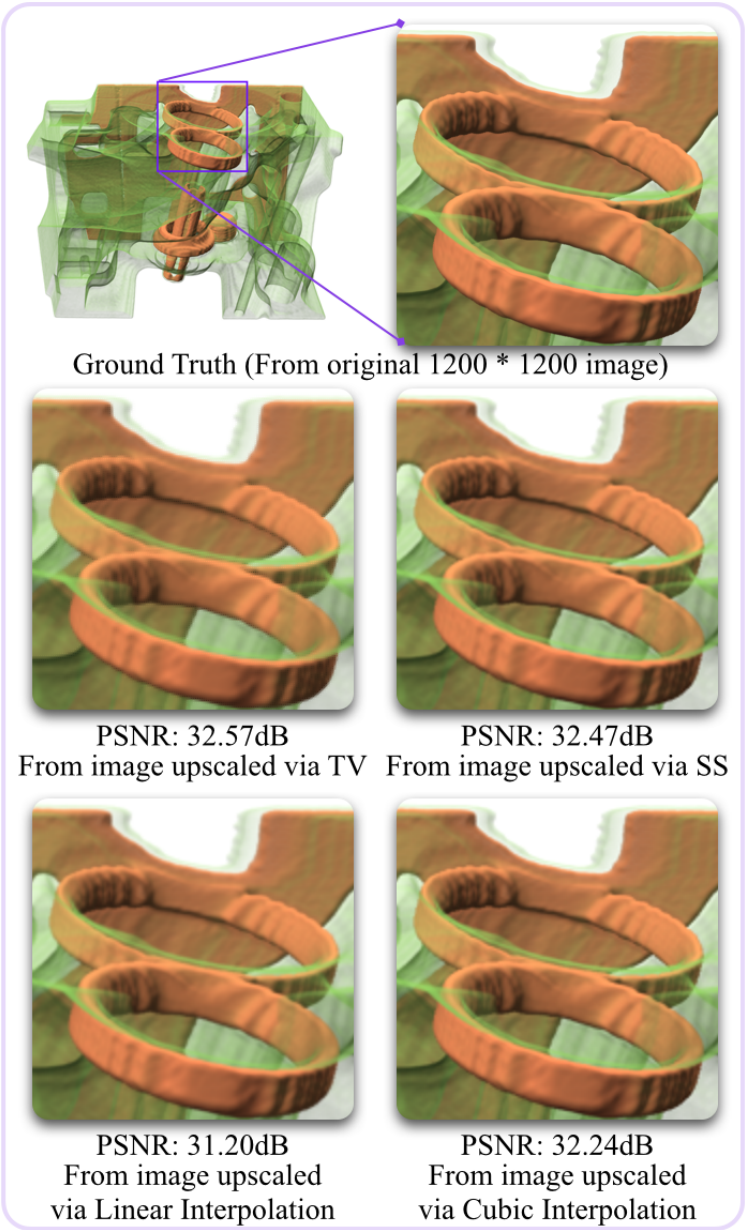


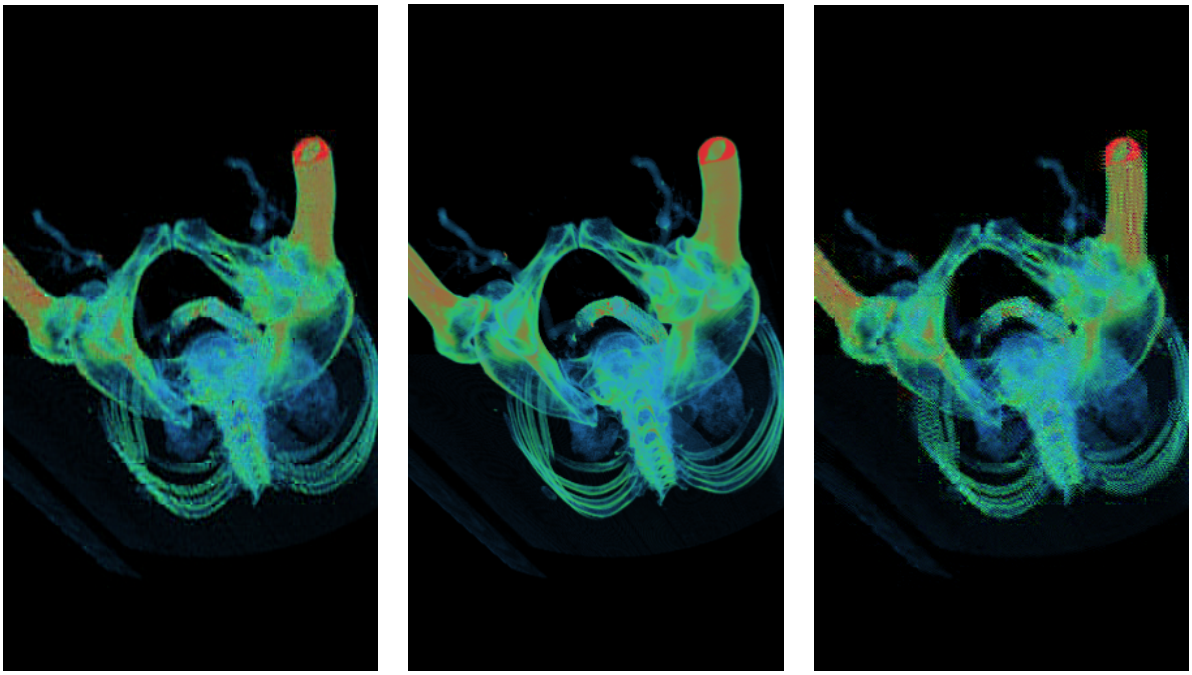
Figure 4.19: Upscaling results via TV and SS

We can also use these methods for image up-scaling, i.e. we can render at one resolution but recover at a higher resolution (also known as super resolution). In this case, we generated an image with higher resolution from a full low resolution image. The pixels from the low resolution image are mapped to the high-resolution image and the missing pixels are recovered. Fig. 4.19 demonstrate the results for this application. The images shown in the second row were upscaled from a low resolution 600×600 image, which is equivalent to the original formulation with 75% missing pixels. We compared the results with those generated with linear interpolation and cubic interpolation Fig. 4.19 and the results are not that different with our methods beating cubic interpolation slightly. It is acceptable since this is just a possible application and the linear and cubic interpolation might work better on this particular pattern, i.e. regularly spaced sample grid. Overall, the results are slightly worse than the case of image recovery at the same resolution (see Fig. 4.13), but the recovery quality remains acceptable (around 30dB).

4.6 Generalizability of Dictionary Learning

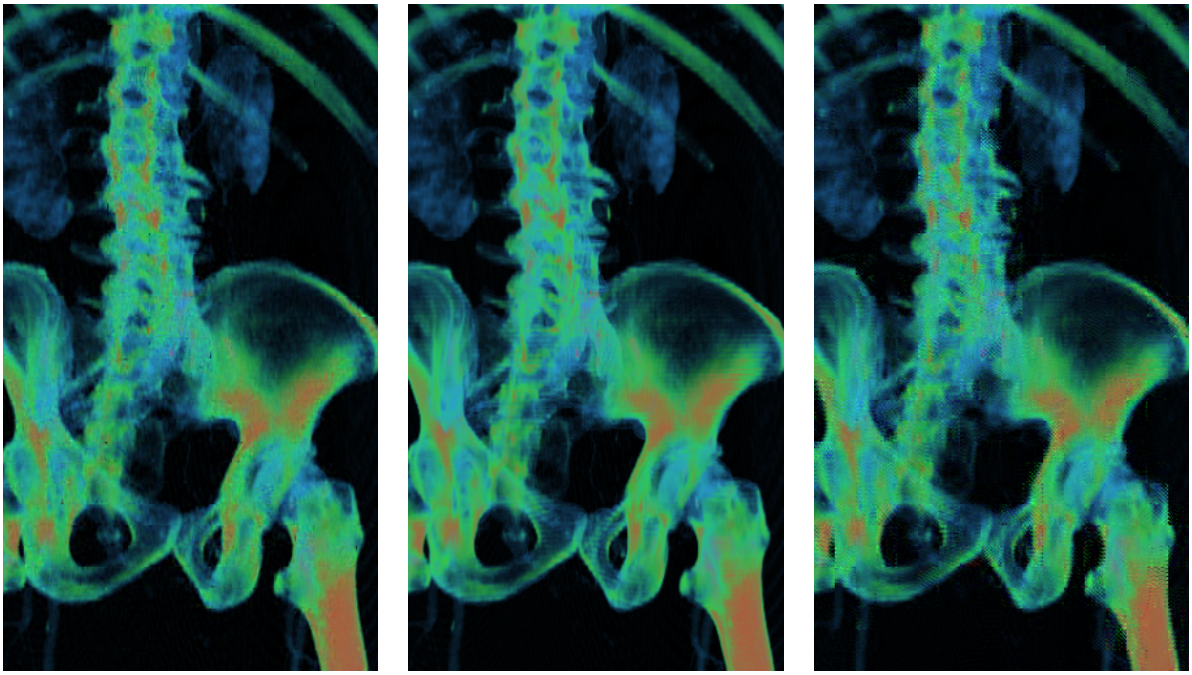
In this section, we discuss the generalizability of our trained dictionary. We used the same trained dictionary as described in Section 4.1.2 to test for the following scenarios and compare it against the DCT dictionary.

Different viewpoint In order to test how well the dictionary responds to changes in viewpoint, we rendered the dataset from a very different viewpoint while keeping the transfer function unaltered. The reconstructed images for the case of 70% missing pixels are shown in Fig. 4.20; a quantitative comparison is shown in Fig. 4.22. The results show that the trained dictionary is robust to viewpoint changes and outperforms the DCT dictionary when more than 50% of the pixels are missing. These quantitative results are very similar to those shown in Fig. 4.8.



(a) Trained, 29.2dB, 4.59s (b) Original (c) DCT, 25.4dB, 6.83s

Figure 4.20: Reconstructing an image rendered from a totally different viewpoint with 70% missing pixels ($L = 30$, encoding times are indicated).



(a) Trained, 30.2dB, 7.86s (b) Original (c) DCT, 26.13dB, 12.39s

Figure 4.21: Reconstructing an image rendered with a zoom factor of 1.5 with 70% missing pixels ($L = 30$, encoding times are indicated).

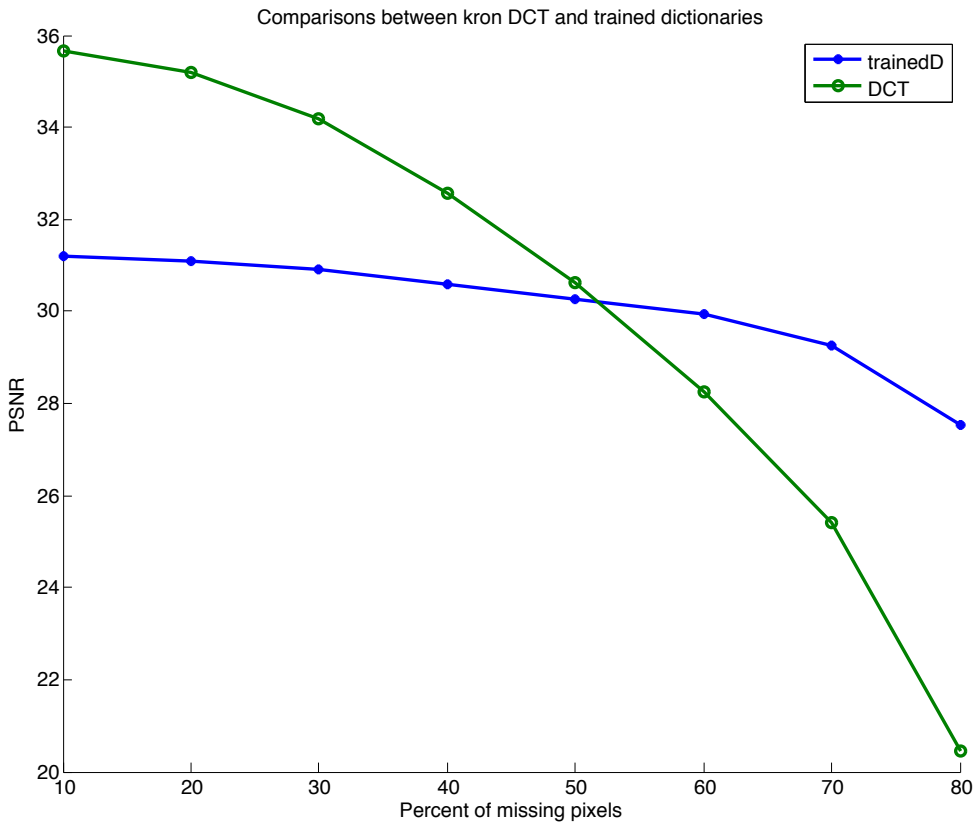
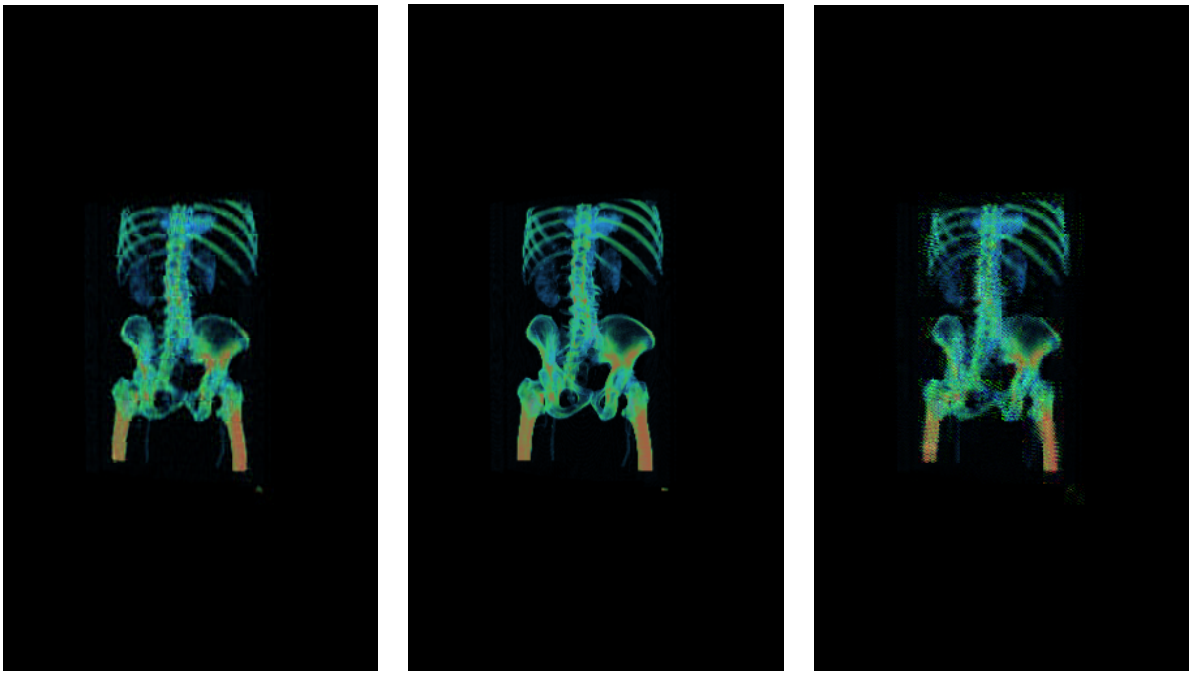


Figure 4.22: Dictionary comparison with changing viewpoint ($L = 30$).

Different Scale In particular situations, users might want to zoom in or zoom out when visualizing a volumetric dataset. Next, we tested this scenario by rendering the test image at two different scales by changing the zoom factor of the camera. As expected, our dictionary can recover from partial information very well as shown in Fig. 4.21 and Fig. 4.23.

Different Transfer Function Next, we compared the two dictionaries when changing the transfer function. For the first test, we just used a simpler piecewise constant transfer function composed of one color and opacity. The case of 70% missing pixels is shown in Fig. 4.24. Since the rendered image is structurally very similar to the test image, we observe that the trained dictionary outperforms the DCT and exhibits reduced block artifacts. The



(a) Trained, 32.5dB, 2.22s (b) Original (c) DCT, 27.5dB, 3.05s

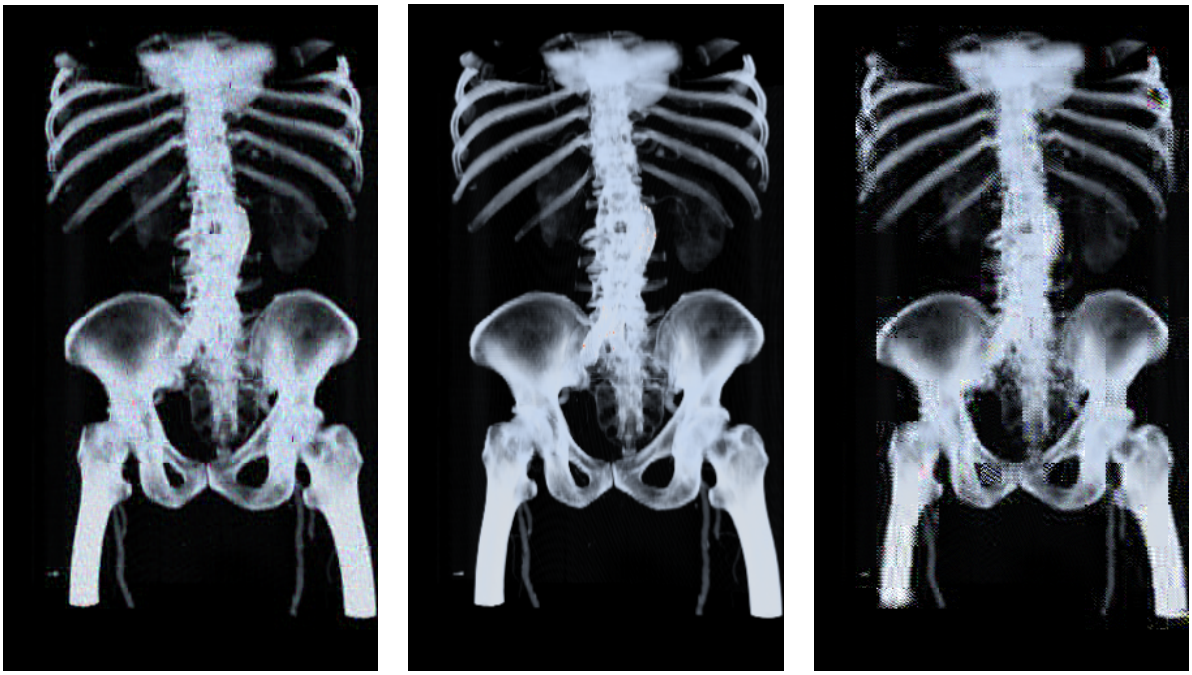
Figure 4.23: Reconstructing and image rendered with a zoom factor of 0.5 with 70% missing pixels ($L = 30$, encoding times are indicated).

encoding step finds the right mix of coefficients needed to blend the atoms of the trained dictionary. Even though the original rendered image is monochromatic, one can observe slight colour fluctuations in both the sparse representations.

For the second test, we increased the opacity of the blue component of the transfer function and also changed the zoom factor. The results are shown in Fig. 4.25 and demonstrate that the trained dictionary is reslient to this change as well and outperforms the DCT.

Different Dataset In order to investigate the generalizability of the trained dictionary to arbitrary datasets, we rendered test images from a CT head dataset and a CT fish dataset. The transfer functions were chosen to reveal images with colors that are different from those present in the training dataset.

Fig. 4.1 shows the original head rendition together with images recovered from sparse representations obtained using the trained and DCT dictionaries (without any missing pixels). For this case, the rendering parameters were chosen so that the original image exhibits

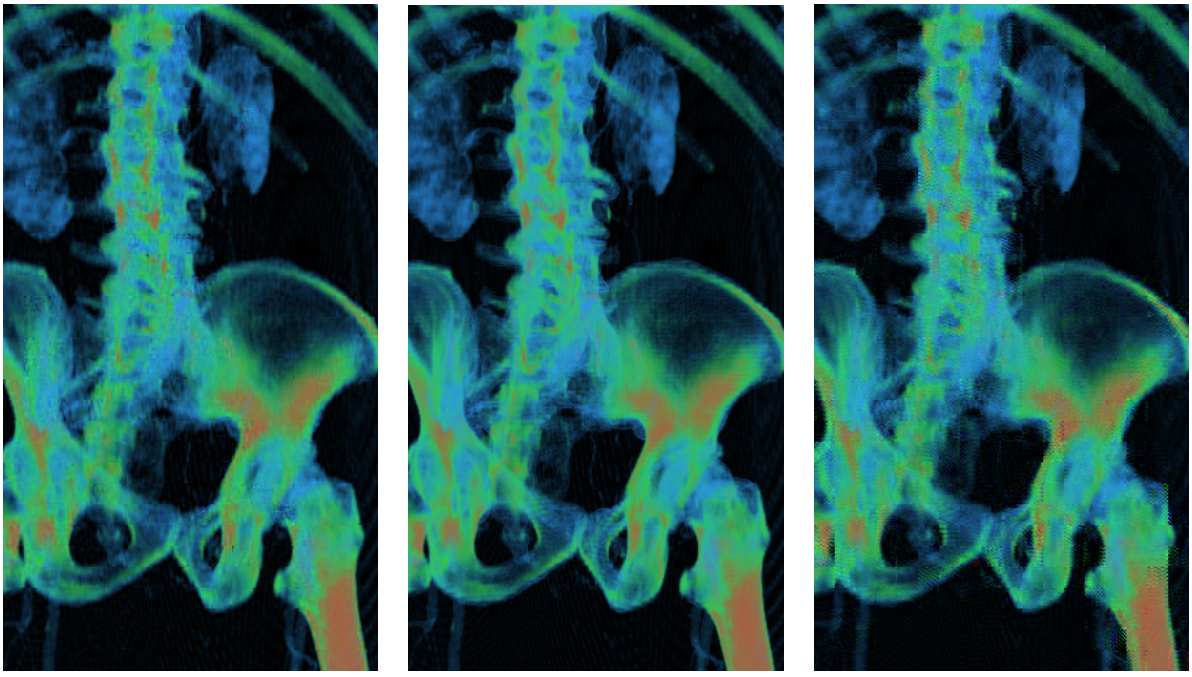


(a) Trained, 27.6dB, 6.51s (b) Original (c) DCT, 22.5dB, 9.77s

Figure 4.24: The effect of changing the transfer function. Images recovered from 70% missing pixels ($L = 30$, encoding times are indicated).

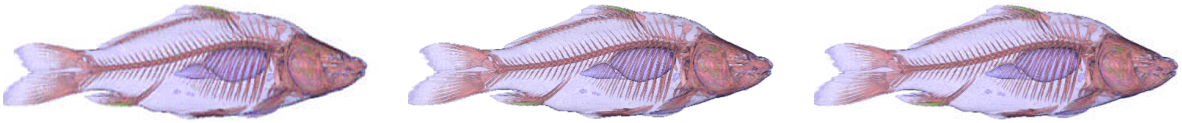
strong ringing artifacts. The DCT dictionary approximates the image well but also reproduces the ringing artefacts present in the original image. In contrast, the trained dictionary recovers an approximation that is somewhat blurred but exhibits much less ringing. This behavior is to be expected since the dictionary was trained on clean images that do not exhibit such artifacts. Image restoration is an added benefit of trained dictionaries.

Fig. 4.26 shows a rendition of the fish dataset encoded using the component-wise DCT dictionary and our trained dictionary (without any missing pixels). Again, we observe that the trained dictionary faithfully recovers the color and structural content of the image even though it uses three times fewer atoms as compared to the DCT dictionary. It is blurred but that should not come as a surprise since the dictionary was trained using an entirely different dataset.



(a) Trained, 30.1dB (b) Original (c) DCT, 25.9dB

Figure 4.25: Changing the opacity and the zoom factor. Results recovered from 70% missing pixels ($L = 30$).



(a) Trained, 30.1dB (b) Original (c) DCT, 25.9dB

Figure 4.26: Recovering sparse approximations of the fish image ($L = 30$).

4.7 Performance Aspects

As discussed in Section 4.1.2, the training process took 47.3 minutes on a single thread implementation running on an Intel®Core™i7-4770K @ 3.50GHz CPU with 16 GB RAM. Those images only have a resolution of 288 × 512 and the training time depends on the number of image blocks of all training data. Therefore, it will increase dramatically as the image resolution increases given that the number of training images stays the same. As mentioned in Section 4.2, for all other methods, test images have a resolution of 1200 ×

1200 and the expected time for dictionary learning will make this approach only practical for offline applications. Fig. 4.9 illustrates the recovery results for color images with different percentages of missing pixels. Looking at the timing results in Table 4.1, we observe that the trained dictionary is about twice as fast when encoding images with 80% missing pixels. However, this advantage is lost when encoding images with 10% missing pixels. The speedup at high fraction of missing pixels is to be expected since the decimated trained dictionary is smaller and recovers three times fewer coefficients. Moreover, the total recovery time is negligible compared to the training procedure.

	Trained			DCT		
	80%	60%	10%	80%	60%	10%
Enc. time (s)	5.86	9.17	20.99	11.11	12.22	16.57
Dec. time (s)	0.57	0.55	0.53	0.17	0.17	0.18
PSNR (dB)	28.5	31.1	32.2	18.8	28.9	36.7

Table 4.1: Timing Comparison for the test image ($L = 30$).

For recovery via other methods, the reconstruction time for different fractions of missing pixels is shown in Fig. 4.27. We reckon that our SS algorithm outperforms all the other methods. This is due to the fact that SS utilizes a least-squares solver which is more efficient as compared to ℓ_1 and TV minimization. CS-Gradients and TV minimization exhibit comparable performance trends while CS-Wavelet lags behind by a wide margin due to the fact that it needs to evaluate the forward and inverse discrete wavelet transforms at every iteration which is slower as compared to the fast Fourier transform.

It should be stressed that these performance results only compare the trends shown by the different algorithms for image recovery. The overall cost of the rendering operation is the sum of the time spent tracing rays and the time it takes to recover the full image from the rendered subset. Since rendering time can vary greatly depending on the quality of the renderer, compressive volume rendering only makes sense when the time spent tracing rays is much greater than the image recovery time. As an example, Table 4.2 shows the total render time for the head and engine datasets using our single-threaded software renderer that

	Head	Engine
600×600	1754.92s	1093.61s
900×900	3955.13s	2442.28s
1200×1200	7067.32s	4249.44s

Table 4.2: Rendering timing for different resolutions.

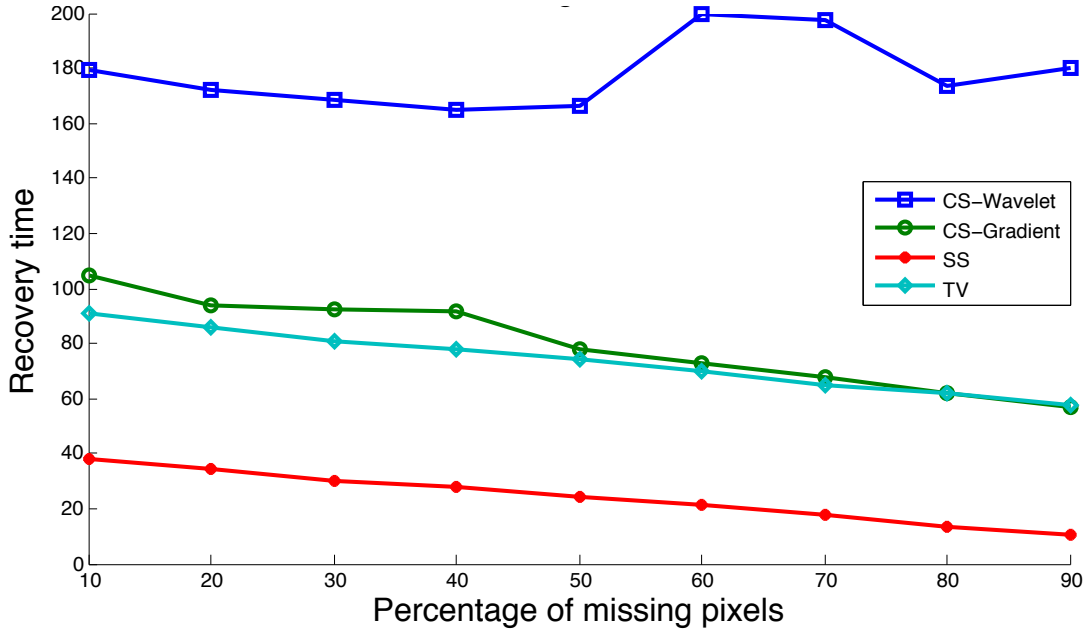


Figure 4.27: Timing results for head data via different recovery methods (1200×1200 images).

uses high-quality tricubic interpolation for both the scalar data and the gradient. Rendering one-fourth of the image pixels cuts down the rendering time by several thousand seconds. The recovery time is only a tiny fraction of this. At the very least, compressive rendering is a viable strategy to accelerate offline volume rendering tasks.

In an interactive environment, one would need to consider additional factors such as the parallelizability of these recovery methods, and their implementation on the GPU. The work of Zach *et al.* [ZPB07] presents a GPU implementation of TV-L1 minimization; they demonstrate real-time optical flow calculation with 30 FPS for a resolution of 320 × 240 pixels. However, they used an iterative conjugate gradient solver to do the TV-L1 minimization whose convergence is not as fast as compared to methods such as NESTA [BBC11]. Least-squares linear systems appear in various problems in Computer Graphics and have a

long histroy of successful GPU acceleration [PF05]. For the SS method, significant speed up can be obtained as the matrix operations in equation 3.21 can be expressed as fast image filtering operations.

Our focus in this work is on investigating the theoretical aspects of the recovery algorithms in the context of volume rendering. With a careful consideration of the inherent parallelizability of these methods, we anticipate significant gains in terms of efficiency or quality. For instance, one could employ a higher quality renderer and only trace a fraction of the rays to achieve a better overall image for the same computational cost. It is also possible to improve reconstruction quality in a two-pass rendering scheme where the first pass recovers a low quality image that can be used to adaptively trace rays according to the content of the image.

4.8 Memory Requirement

It is also important to discuss the memory requirements of all methods since memory consumption can be one factor which needs to be taken care of in real applications.

Among all our methods, recovery via dictionary learning is the only one that needs extra memory to perform the task. One has to train the dictionary properly from the training datasets and store followed by transmitting it when needed. In a client-server scenario, the cloud-based server can perform training, rendering and the encoding steps and transmit the trained dictionary and the sparse coefficients to the client. Thus, extra memory is needed to store the dictionary locally and attentions have to be drawn especially when the size of dictionary is large.

In contrast, all other methods do not have memory concerns since they just take the images with missing pixels as input and produce the recovery without storing anything.

Chapter 5

Conclusion

5.1 Summary

We presented four different methods for recovering images from a subset of the pixels. Our results show that the CS-based approaches are not suitable for this problem as we are restricted to making pixel measurements. The previously proposed approach (CS-Wavelet) exhibits strong artefacts when the fraction of missing pixels in high. Our attempt to improve the method (CS-Gradient) only shows slight improvements.

Moreover, we used the K-SVD algorithm [MAB06] to train greylevel and color dictionaries based on exemplar volume rendered images. The training images are very simple to generate and require no preprocessing. Yet, the learned dictionary is resilient to changes in viewpoint and scale and can faithfully recover images from a small set of rendered pixels. We compared our trained dictionary with a preconstructed DCT dictionary and demonstrated that the trained dictionary achieves higher sparsity levels and can potentially be used for denoising and artifact removal. Furthermore, we also tested the generalizability of the trained dictionary and got acceptable results that are inferior to the DCT dictionary. Our experimental results show that a dictionary based on the DCT, while being more general, is not as efficient as a trained dictionary. On the other hand, a trained dictionary achieves higher compression ratios, is more robust in the presence of missing data, and is also generalizable, thus making it suitable for various volume visualization applications. However, the recovery quality is not consistent especially when the percentage of missing pixels is high while the training phase is relatively long. Thus, in terms of sparsity level, recovery via dictionary learning is worth exploring for volume rendered images.

On the other hand, smoothness-based methods (TV and SS) are much more suitable for

Image	Category	TV/SS with LD
Engine, Head and Foot (Fig. 4.13, Fig. 4.16)	DVR: very smooth image.	Consistent recovery results over different fractions of missing pixels.
Watch (Fig. 4.18)	Physically-based rendering: smooth image, very few small scale details.	Good recovery results even with high fractions of missing pixels.
Aneurysm (Fig. 4.16)	ISR: not so smooth, some small scale details and sharp edges.	Acceptable recovery with about 50% missing pixels but some degradation when the fraction of missing pixels is high.
Synthetic (Fig. 4.15)	Non smooth: lots of small scale details and edges.	Unacceptable recovery even with a very small fraction of missing pixels.

Table 5.1: Performance summary of TV and SS for different types of images.

this task and are able to recover images successfully even when the fraction of missing pixels is high. A summary of the recovery results based on the smoothness-inspired methods is shown in Table 5.1. For performance considerations, we advocate the SS method as it is based on a least-squares solver.

5.2 Contributions

In this thesis, we focused on recovering direct volume rendered images from images rendered with a small subset of pixels and we investigated four different methods which can speed up the volume rendering process while maintaining great image quality. Specifically this thesis has shown that:

1. Compressed Sensing (CS) based approaches (CS-Wavelet and CS-Gradient) cannot produce consistent recovery for different percentage of missing pixels due to incoherence issues (CS-Wavelet) as well as restrictions on making pixel measurements (CS-Gradient).

2. Recovery via dictionary learning can produce sparse representations with high

level of sparsity, which is good in terms of compressibility. However, it imposes limitations on recovery quality when the percentage of missing pixels is high as well as the long training process makes it unsuitable for this compressive volume rendering task.

3. By comparing and contrasting recovery results via different methods, we empirically showed that smoothness-based methods(TV and SS) are more suitable for recovering images from a fraction of missing pixels, especially when the percentage of missing pixels is high.

5.3 Future Work

For recovery via dictionary learning, we believe that the generalizability of the dictionary can be improved by incorporating multiscale learning techniques [MSE07, MES08]. In our reconstruction phase, we have encoding and decoding parts. The cost of the decoding part is insignificant as compared to the encoding part. For our color dictionary, the encoding part can be up to twice as fast in the presence of missing pixels. All of our experiments were performed with a single CPU thread. An important point is to consider the fraction of the overall rendering time that the encoding/decoding phase takes. A single-threaded volume renderer can be quite expensive depending on the rendering parameters used (step size, interpolation, illumination, multisampling etc). In contrast, the encoding/decoding phase is in the image domain and is independent of the particular rendering method used. Furthermore, its main ingredient is OMP which is highly parallelizable as recently confirmed by a CUDA implementation that has reported thirty to forty-fold speedups on a middle class GPU [FCWH11].

We would also like to comment on the possible usage scenarios where overcomplete image-space dictionaries can potentially be useful. Owing to the high level of sparsity that these dictionaries can attain, cloud-based visualization environments are an ideal deployment sce-

nario. For instance, a cloud-based server can perform the rendering and encoding steps and transmit the sparse coefficient vector to a thin client such as a tablet or a phone. Users who need to visualize a large collection of volumetric datasets that are similar to each other can greatly benefit from dictionary learning. A dictionary can be trained using a collection of training images generated with a preconfigured transfer function and representative viewpoints. As our results demonstrate, such images can readily be recovered from a small fraction of the pixels with a high fidelity, and are also resilient to changes in scale and viewpoint.

Recovery via smoothness-based methods can provide consistent results over different fractions of missing pixels. In future, besides exploring the matrix and tensor completion methods mentioned earlier, we are also interested in further comparing and contrasting the smoothness-based methods in an interactive volume rendering environment, both in terms of performance and in terms of quality with respect to perceptual metrics. A careful treatment of parallelizability and GPU implementation is a subject of future work and significant speed-up can therefore be expected for this type of problem if the inherent parallelism is properly exploited.

Bibliography

[AHPV05] Agarwal P. K., Har-Peled S., Varadarajan K. R.: Geometric approximation via coresets. *Combinatorial and computational geometry 52* (2005), 1–30.

[And93] Anderson P. G.: Linear pixel shuffling for image processing: an introduction. *Journal of Electronic Imaging 2*, 2 (1993), 147–154.

[ASHU05] Arigovindan M., Sühling M., Hunziker P., Unser M.: Variational image reconstruction from arbitrarily spaced samples: A fast multiresolution spline solution. *IEEE Transactions on Image Processing 14*, 4 (April 2005), 450–460.

[BBC11] Becker S., Bobin J., Candès E. J.: NESTA: a fast and accurate first-order method for sparse recovery. *SIAM Journal on Imaging Sciences 4*, 1 (2011), 1–39.

[BHP14] Beyer J., Hadwiger M., Pfister H.: A survey of gpu-based large-scale volume visualization. *Proceedings EuroVis 2014)* (2014).

[BMWM06] Bergner S., Möller T., Weiskopf D., Muraki D. J.: A spectral analysis of function composition and its implications for sampling in direct volume visualization. *Visualization and Computer Graphics, IEEE Transactions on 12*, 5 (2006), 1353–1360.

[BSCB00] Bertalmio M., Sapiro G., Caselles V., Ballester C.: Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 417–424.

[BTU01] Blu T., Thcvenaz P., Unser M.: Moms: Maximal-order interpolation of minimal support. *Image Processing, IEEE Transactions on 10*, 7 (2001), 1069–1080.

[Buh00] Buhmann M. D.: Radial basis functions. *Acta Numerica 2000 9* (2000), 1–38.

[CBL89] CHEN S., BILLINGS S. A., LUO W.: Orthogonal least squares methods and their application to non-linear system identification. *International Journal of Control 50*, 5 (1989), 1873–1896. URL: http://www.tandfonline.com/doi/abs/10.1080/00207178908953472, arXiv:http://www.tandfonline.com/doi/pdf/10.1080/00207178908953472, doi:10.1080/00207178908953472.

[CDS01] Chen S., Donoho D., Saunders M.: Atomic decomposition by basis pursuit. *SIAM Review 43*, 1 (2001), 129–159. URL: http://epubs.siam.org/doi/abs/10.1137/S003614450037906X, arXiv:http://epubs.siam.org/doi/pdf/10.1137/S003614450037906X, doi:10.1137/S003614450037906X.

[CEPY05] Chan T., Esedoglu S., Park F., Yip A.: Recent developments in total variation image restoration. In *In Mathematical Models of Computer Vision* (2005), Springer Verlag.

[CR09] Candès E. J., Recht B.: Exact matrix completion via convex optimization. *Foundations of Computational mathematics 9*, 6 (2009), 717–772.

[CS07] Czumaj A., Sohler C.: Sublinear-time approximation algorithms for clustering via random sampling. *Random Structures and Algorithms 30*, 1-2 (2007), 226–256. URL: http://dx.doi.org/10.1002/rsa.20157, doi:10.1002/rsa.20157.

[CT06] Candes E. J., Tao T.: Near-optimal signal recovery from random projections: Universal encoding strategies? *Information Theory, IEEE Transactions on 52*, 12 (2006), 5406–5425.

[DMA97] Davis G., Mallat S., Avellaneda M.: Adaptive greedy approximations. *Constructive approximation 13*, 1 (1997), 57–98.

[DMZ94] Davis G. M., Mallat S. G., Zhang Z.: Adaptive time-frequency decompositions. *Optical Engineering 33*, 7 (1994), 2183–2191. URL: `http://dx.doi.org/10.1117/12.173207`, `doi:10.1117/12.173207`.

[EAH00] Engan K., Aase S. O., Husøy J. H.: Multi-frame compression: Theory and design. *Signal Processing 80*, 10 (2000), 2121–2140.

[EK12] Eldar Y., Kutyniok G.: *Compressed Sensing: Theory and Applications*. Cambridge University Press, 2012.

[Ela10] Elad M.: *Sparse and redundant representations: from theory to applications in signal and image processing*. Springer, 2010.

[FCWH11] Fang Y., Chen L., Wu J., Huang B.: GPU implementation of orthogonal matching pursuit for compressive sensing. In *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on* (Dec 2011), pp. 1044–1047. `doi:10.1109/ICPADS.2011.158`.

[GIGM12] Gobbetti E., Iglesias Guitián J. A., Marton F.: Covra: A compression-domain output-sensitive volume rendering architecture based on a sparse representation of voxel blocks. In *Computer Graphics Forum* (2012), vol. 31, Wiley Online Library, pp. 1315–1324.

[GL07] Guo K., Labate D.: Optimally sparse multidimensional representation using shearlets. *SIAM journal on mathematical analysis 39*, 1 (2007), 298–318.

[GO09] Goldstein T., Osher S.: The split bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences 2*, 2 (2009), 323–343.

[GR97] Gorodnitsky I., Rao B.: Sparse signal reconstruction from limited data using focuss: a re-weighted minimum norm algorithm. *Signal Processing, IEEE Transactions on 45*, 3 (1997), 600–616. doi:10.1109/78.558475.

[HKRs*06] Hadwiger M., Kniss J. M., Rezk-salama C., Weiskopf D., Engel K.: *Real-time Volume Graphics*. A. K. Peters, Ltd., Natick, MA, USA, 2006.

[HSS*05] Hadwiger M., Sigg C., Scharsach H., Bühler K., Gross M.: Real-time ray-casting and advanced shading of discrete isosurfaces. In *Computer Graphics Forum* (2005), vol. 24, Wiley Online Library, pp. 303–312.

[KDMR*03] Kreutz-Delgado K., Murray J. F., Rao B. D., Engan K., Lee T.-W., Sejnowski T. J.: Dictionary learning algorithms for sparse representation. *Neural computation 15*, 2 (2003), 349–396.

[LGBB05] Lesage S., Gribonval R., Bimbot F., Benaroya L.: Learning unions of orthonormal bases with thresholded singular value decomposition. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on* (2005), vol. 5, IEEE, pp. v–293.

[LLYM04] Ljung P., Lundstrom C., Ynnerman A., Museth K.: Transfer function based adaptive decompression for volume rendering of large medical data sets. In *Volume Visualization and Graphics, 2004 IEEE Symposium on* (2004), IEEE, pp. 25–32.

[LMWY13] Liu J., Musialski P., Wonka P., Ye J.: Tensor completion for estimating missing values in visual data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 35*, 1 (2013), 208–220.

[MAB06] M. Aharon M. E., Bruckstein A.: K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representation. In *IEEE Trans. Signal Process.* (2006), vol. 54, pp. 4311–4322.

[MES08] Mairal J., Elad M., Sapiro G.: Sparse representation for color image restoration. *Image Processing, IEEE Transactions on 17*, 1 (2008), 53–69.

[MGBB00] Marcellin M. W., Gormish M. J., Bilgin A., Boliek M. P.: An overview of jpeg-2000. In *Data Compression Conference, 2000. Proceedings. DCC 2000* (2000), IEEE, pp. 523–541.

[MKD01] Murray J. F., Kreutz-Delgado K.: An improved focuss-based learning algorithm for solving sparse linear inverse problems. In *Signals, Systems and Computers, 2001. Conference Record of the Thirty-Fifth Asilomar Conference on* (2001), vol. 1, IEEE, pp. 347–351.

[MP10] Ma J., Plonka G.: The curvelet transform. *Signal Processing Magazine, IEEE 27*, 2 (2010), 118–133.

[MSE07] Mairal J., Sapiro G., Elad M.: Multiscale sparse image representation with learned dictionaries. In *IEEE International Conference on Image Processing (ICIP 2007)* (2007), vol. 3, IEEE, pp. III–105.

[MZ93] Mallat S., Zhang Z.: Matching pursuits with time-frequency dictionaries. *Signal Processing, IEEE Transactions on 41*, 12 (1993), 3397–3415. doi:10.1109/78.258082.

[NW13] Needell D., Ward R.: Stable image reconstruction using total variation minimization. *SIAM Journal on Imaging Sciences 6*, 2 (2013), 1035–1058.

[PF05] Pharr M., Fernando R.: *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems).*

Addison-Wesley Professional, 2005.

[PGB03] Pérez P., Gangnet M., Blake A.: Poisson image editing. In *ACM Transactions on Graphics (TOG)* (2003), vol. 22, ACM, pp. 313–318.

[PH10] Pharr M., Humphreys G.: *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2010.

[PMGC12] Patel V. M., Maleh R., Gilbert A. C., Chellappa R. a.: Gradient-based image recovery methods from incomplete fourier measurem ents. *IEEE Transactions on Image Processing 21*, 1 (2012), 94–105.

[PML*09] Peers P., Mahajan D. K., Lamond B., Ghosh A., Matusik W., Ramamoorthi R., Debevec P.: Compressive light transport sensing. *ACM Transactions on Graphics (TOG) 28*, 1 (2009), 3.

[PRK93] Pati Y., Rezaiifar R., Krishnaprasad P. S.: Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on* (1993), pp. 40–44 vol.1. doi:10.1109/ACSSC.1993.342465.

[RBE10] Rubinstein R., Bruckstein A. M., Elad M.: Dictionaries for sparse representation modeling. *Proceedings of the IEEE 98*, 6 (2010), 1045–1057.

[RGG*12] Rodríguez M. B., Gobbetti E., Guitián J. A. I., Makhinya M., Marton F., Pajarola R., Suter S. K.: A survey of compressed gpu-based direct volume rendering. In *Eurographics 2013-State of the Art Reports* (2012), The Eurographics Association, pp. 117–136.

[RKZ12] Rousselle F., Knaus C., Zwicker M.: Adaptive rendering with non-local means filtering. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 195:1–195:11.

[ROF92] Rudin L. I., Osher S., Fatemi E.: Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena 60*, 1 (1992), 259–268.

[SD09] Sen P., Darabi S.: Compressive dual photography. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 609–618.

[SD11] Sen P., Darabi S.: Compressive rendering: A rendering application of compressed sensing. *Visualization and Computer Graphics, IEEE Transactions on 17*, 4 (2011), 487–499.

[Squ07] Squillacote A.: *The ParaView Guide*. Kitware, 2007. URL: `http://books.google.ca/books?id=mJACPwAACAAJ`.

[TF11] Tošić I., Frossard P.: Dictionary learning. *IEEE Signal Processing Magazine 28*, 2 (March 2011), 27–38. `doi:10.1109/MSP.2010.939537`.

[TG07] Tropp J. A., Gilbert A. C.: Signal recovery from random measurements via orthogonal matching pursuit. *Information Theory, IEEE Transactions on 53*, 12 (2007), 4655–4666.

[TL93] Totsuka T., Levoy M.: Frequency domain volume rendering. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (1993), ACM, pp. 271–278.

[TM] Taubman D., Marcellin M.: *JPEG2000 Image Compression Fundamentals, Standards and Practice: Image Compression Fundamentals, Standards, and Practice*. JPEG2000: Image Compression Fundamentals, Standards, and Practice.

[Tro04] Tropp J.: Greed is good: algorithmic results for sparse approximation. *Information Theory, IEEE Transactions on 50*, 10 (2004), 2231–2242. `doi:10.1109/TIT.2004.834793`.

[WAG*12] Wenger S., Ament M., Guthe S., Lorenz D., Tillmann A., Weiskopf D., Magnor M.: Visualization of astronomical nebulae via distributed multi-gpu compressed sensing tomography. *Visualization and Computer Graphics, IEEE Transactions on 18*, 12 (2012), 2188–2197.

[WBSS04] Wang Z., Bovik A. C., Sheikh H. R., Simoncelli E. P.: Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on 13*, 4 (2004), 600–612.

[Wik15a] Wikipedia: Inpainting — Wikipedia, the free encyclopedia, 2015. [Online; accessed 17-March-2015]. URL: `http://en.wikipedia.org/wiki/Inpainting`.

[Wik15b] Wikipedia: Volume rendering — Wikipedia, the free encyclopedia, 2015. [Online; accessed 04-June-2015]. URL: `http://en.wikipedia.org/wiki/Volume_rendering`.

[Wik15c] Wikipedia: Wavelet transform — Wikipedia, the free encyclopedia, 2015. [Online; accessed 05-June-2015]. URL: `http://en.wikipedia.org/wiki/Wavelet_transform`.

[XAE12] Xu X., Alvarado A. S., Entezari A.: Reconstruction of irregularly-sampled volumetric data in efficient box spline spaces. *Medical Imaging, IEEE Transactions on 31*, 7 (2012), 1472–1480.

[XSE14] Xu X., Sakhaee E., Entezari A.: Volumetric data reduction in a compressed sensing framework. *Proceedings EuroVis 2014)* (2014).

[ZPB07] Zach C., Pock T., Bischof H.: A duality based approach for realtime tv-l 1 optical flow. In *Pattern Recognition*. Springer, 2007, pp. 214–223.