Adaptive Regularization for Inverse Problems in Imaging

by

Cameron J. Blocker

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical and Computer Engineering)
in the University of Michigan
2022

Doctoral Committee:

Professor Jeffrey A. Fessler, Chair
Associate Professor Laura Balzano
Assistant Professor David F. Fouhey
Associate Professor Raj Rao Nadakuditi
Chief Scientist Richard G. Paxman, Maxar Technologies

Cameron Blocker

cblocker@umich.edu

ORCID iD: 0000-0003-3818-4127

ACKNOWLEDGMENTS

It would be impossible for me to enumerate the many people who have influenced my path to this day. Teachers, professors, friends, mentors, work associates, classmates, church leaders, family and more have impacted my goals and how I think. I would not be where I am today if it was not for their many helping hands. Indeed, I would not be who I am without them and I feel the utmost gratitude to have been so fortunate. As a more feasible task, here I would like to express that gratitude to the people who directly made this thesis possible these last several years.

First of all, I would like to thank my advisor, Jeff Fessler. Thank you for believing in me and encouraging me in my research, from when I was first applying and was indecisive to throughout my many roadblocks till today. Thank you for your patience and your wisdom. I could not have hoped for a better mentor, and I will continue to aspire to your level of intellect, creativity and understanding. You have had the most significant impact of anyone on how I think about research problems and how I approach solving them. Thank you for helping me to develop as a researcher by providing me with both a generous amount of support and independence. It has been an honor to be your student. A paragraph could never do your contributions these past six years justice.

I am grateful to each of my committee members for their generous contribution of time and support. I selected each of them for my committee because I admired their research and, for many of them, have had the opportunity to work with them either as a research collaborator, a student of one of their classes, or a graduate student instructor for their class. They have each made a unique contribution to this thesis and to me as a researcher. Thank you!

In addition to my advisor, I have had many coauthors that have made my published research possible. I am grateful for the opportunity to work with each of them on such exciting projects and the insights they have provided me. I appreciate the generous funding from the University of Michigan, the Keck Foundation, NIH, NSF, and KLA that made this research possible. This thesis only highlight a few of the projects I worked on. None would have been possible without my coauthors and funding.

Next, I would like to thank the many students and post-docs in the "Lab of Jeff" and adjacent groups who have enriched my Ph.D. experience from when I first arrived in 2016 till today. They have provided a forum for mentorship and engaging technical discussion that has exposed me to more ideas than I ever could have come across on my own. In addition to the technical strengths they provide, they have been a great source of empathy through what can be a challenging degree. Thank you for celebrating my successes and commiserating with me in the rough times. After witnessing so many Ph.D. students pass through the group, it is hard to believe it is now my turn to graduate and I look forward to hearing about the students who still remain.

Finally, I am thankful for my family, including my fianceé. My family has been an inexhaustible source of support for me throughout the years. Thank you, Makayla, for helping to keep me going and for all of the great memories these last few years. And most of all, thank you to my dear parents, who have have nurtured and provided for my interest in science and technology for as long as I can remember. Thank you for your unceasing love and countless prayers on my behalf. I could never have made it here without you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

LIST OF ACRONYMS

MBIR model-based image reconstruction

MAP maximum a posteriori

SVD singular value decomposition

PCA principle component analysis

AWGN additive white Gaussian noise

SAI subaperture image

EPI epipolar image

LF light field

CFA color filter array

MRI magnetic resonance imaging

CT computerized tomography

UTL unitary transform learning

SOUP sum of outer products

BCD block coordinate descent

MM majorize minimize

CNN convolutional neural network

GAN generative adversarial network

DIP deep image prior

DCT discrete cosine transform

DFT discrete Fourier transform

FFT fast (discrete) Fourier transform

TV Total Variation

PSNR peak signal-to-noise ratio

NRMSE normalized root mean square error

NRMSD normalized root mean square difference

SNR signal-to-noise ratio

ADMM alternating direction method of multipliers

OSSI oscillating steady state imaging

LIST OF SYMBOLS

$\boldsymbol{x}$ a signal or image

$\hat{\boldsymbol{x}}$ an image estimate

$\boldsymbol{x}^*$ a minimizer of a function

$\boldsymbol{y}$ measurements

$\boldsymbol{A}$ the measurement/system/forward model

$\boldsymbol{\epsilon}$ additive white Gaussian noise

$\boldsymbol{e}_i$ ith standard basis vector

$\|\cdot\|_p$ a vector p-norm

$|||\cdot|||_p$ an operator or matrix norm

$\nabla\cdot$ the gradient operator

$\nabla^2\cdot$ the Hessian operator

$\mathsf{R}(\cdot)$ a regularization function

B' Hermitian transpose of B

B^T Transpose of B

$\boldsymbol{T}$ a transform or analysis operator

$\boldsymbol{D}$ a dictionary or overcomplete "basis"

$\mathcal{R}(\cdot)$ the range space/set of a function

$\mathcal{N}(\cdot)$ the null space/set of a function

$\mathrm{prox}_f(\boldsymbol{v}, \gamma)$ proximal operator, $\arg\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{v}-\boldsymbol{x}\|_2^2 + \gamma f(\boldsymbol{x})$

$\mathcal{P}_C(\boldsymbol{v})$ projection onto set C, $\arg\min_{\boldsymbol{x}\in C} \frac{1}{2}\|\boldsymbol{v}-\boldsymbol{x}\|_2^2$

ABSTRACT

We live in a world where imaging systems are ubiquitous. From the cell phones in our pockets to our cars and doorbells and on to telescopes and medical scanners, imaging has changed how we share, document, and understand our world. There is an increasing demand to make these systems more efficient by producing higher-quality images with fewer and fewer resources. When we need an image that is of higher quality than can be directly constructed from its samples, we have an inverse problem and must rely on nonlinear reconstruction schemes. These reconstruction algorithms fill the gap between the measured samples and the desired high-quality images by leveraging a model that attempts to distinguish image content from artifacts.

Many modern image models are optimized — or trained — to produce high-quality results on a large dataset of images. These methods have shown a large improvement over classical fixed models, but require a lot of computation and high-quality data for the dataset. This thesis applies and develops adaptive image models which are trained during reconstruction for a particular image and can optionally be pre-trained on a dataset. These adaptive methods are formulated as part of a regularization term of a minimization problem and enabled by efficient model updates. In particular, we further develop a union of subspaces model and a dynamic subspace model which are both enabled by an efficient generalized Procrustes update. Beyond subspace models, this work also investigates the application of neural networks trained at reconstruction time on image patches. The proposed methods are applied to problems in light-field imaging and MRI.

CHAPTER 1

Introduction

Modern imaging systems rely on computational techniques to produce high-quality images. These systems are often resource-constrained and inherently lossy. For example, CT imaging is limited in how much radiation it can expose the patient to, MRI is limited by the long duration of the scan, and optical imaging systems are constrained by how many photons each pixel is able to collect. All of these systems are further subject to noise and other system nonidealities. These physical constraints limit the number and quality of the measured image samples and are unavoidable at a hardware level.

Computational imaging allows us to go beyond these fixed sampling budgets and system nonidealities to produce higher-quality images than would be possible with the hardware alone. Because these physical degradations cannot be directly removed, or *inverted*, with knowledge of the imaging system alone, reconstructing higher-quality images from lower-quality samples is known as an *inverse problem*.

Solving these inverse problems is the central goal of this work. A common way to handle these inverse problems is to pose the reconstruction as an optimization problem consisting of a *data-fidelity* term encouraging our reconstruction to be consistent with the measured samples and a *regularization* term, that enforce our measurement-independent expectation about the class of images we are reconstructing. The limited nature of the measurements generally makes solving the data-fidelity term alone ill-posed. As such, developing effective regularization is essential to image-reconstruction accuracy. Developing effective regularization is challenging, as it must provide a model that is both flexible enough to represent all plausible true images, yet discriminating enough to reject noise and artifacts.

A common approach in the design of regularization is modeling redundant features with reduced dimensionality. An image, for example, may have repeating or similar textures that, when extracted into smaller image chunks or patches, can be more easily modeled, such as belonging to a union of subspaces. Many signals of interest fall into a shift-invariant class (at least locally). Shifting an image by a few pixels generally produces an

image that is equally realistic. By modeling overlapping image patches instead of entire images, we implicitly model the shift-invariant structure of images.

Modern image regularization is often data-driven. These image models are not fixed, but instead describe a class of image models that we can choose from by optimizing a loss on a dataset of high-quality images. These classes range from the relatively simple, such as subspace models, to the nonlinear and complex, such as neural networks. Often, optimizing these image models can be computationally intensive, in part due to the large size of the dataset. Generally, once these models are trained, they are fixed as part of the regularization term during image reconstruction.

In this work, we apply and develop *adaptive* image models as regularization for solving inverse problems. Like traditional data-driven models, these adaptive models are not fully-fixed and instead describe a class of possible regularizers we can optimize over. As such, these methods can optionally be pre-trained on a dataset. Unlike traditional data-driven regularization though, these models are not fixed at reconstruction time and we continue to optimize over the class of image models as we iterate our reconstruction. Because these methods do not require knowledge of ground-truth images to optimize their parameters, they are also known as "blind" or "instance-adaptive" regularization [65]. As such, they provide an accessible alternative in the absence of a large high-quality dataset.

Theoretically, any data-driven regularizer could be applied adaptively by optimizing its parameters at reconstruction time. But if this class of image models is too broad and expressive, it can easily learn to fit noise and artifacts. Additionally, some data-driven regularizers are computationally intensive to update and so are less favorable to apply in a time and resource-constrained reconstruction method. Thus, a good adaptive regularization is both (1) constrained enough to reject noise and artifacts, and (2) employs an efficient update for its parameters.

The adaptive version of a data-driven regularizer is also subtly different. For example, while a data-driven transform sparsity regularizer will encourage the reconstructed image to be sparse under a particular operator, an adaptive transform sparsity regularizer merely encourages the reconstructed image to be sparsifiable under a broad class of operators. Similarly, a pre-trained convolutional neural network constrains a reconstructed image to lie in or near its range, while an untrained convolutional network such as DIP [82] merely biases the reconstructed image towards smoothness.

1.1 Contributions

This work investigates the applicability of, expands existing, and develops new adaptive regularization methods for inverse problems in imaging. In particular, we investigate the effectiveness of blind unitary transform learning (UTL) for reconstructing light-field images and determine an effective approach for extracting patches from these high-dimensional signals [10]. We then further expand this approach with adaptively set sparsity thresholds by appropriately weighting the sparsity-encouraging regularization. Empirically, we show that blind UTL performs better than pretrained UTL on a heterogeneous dataset. Next, we investigate applying generative neural networks on image patches as both pre-trained and instance-adaptive regularization and compare these to the more constrained UTL method. Finally, we develop a new signal model based on time-varying subspaces on piecewise geodesics [11]. We show this model effectively fits patches of dynamic MRI and video.

1.2 Outline

To start, Chapter 2 provides the necessary background on the mathematical framework we will use for modeling inverse problems. It introduces the details of compressive light-field imaging and MRI as applications of solving inverse problems. We then review transform learning, a specific adaptive regularization.

Chapter 3 builds on the application and regularization of the previous chapter by investigating the effective application of unitary transform learning to light-field reconstruction from focal-stack data. In particular, we investigate how to best distribute the patch dimensions among the light field dimensions.

Chapter 4 further develops this work on UTL by investigating a novel transform weighting scheme which adaptively set sparsity thresholds for blind reconstruction of light fields. Several heuristics for setting these weights are explored and evaluated on light fields jointly reconstructed and demosaiced from raw focal-stack images.

Chapter 5 looks at the generalizability of unitary transform learning on heterogeneous data. We investigate empirically how an instance-adaptive regularizer compares to pretrained regularizer on heterogeneous MRI data.

In Chapter 6, we look beyond sparse subspace models and investigate the use of generative neural networks as regularization on a patch level. We investigate both the use of pretrained networks and untrained networks trained at reconstruction time. The proposed

methods are evaluated on both synthetic phantoms and retrospectively undersampled real MRI data.

Finally, Chapter 7 develops a novel piecewise geodesic model for dynamic subspace estimation. The proposed model is minimized by alternating majorize-minimize updates. The effectiveness of the approach is shown on fMRI data and videos.

Chapter 8 concludes this work. and presents some directions for future work.

In the appendix, we provide more in depth algorithmic detail than is provided in the main chapters. In particular, Appendix A derives a discretized focal stack forward model for light fields using principles of geometric optics.

Appendix B provides supplemental detail for Chapter 7 on dynamic subspace estimation. In particular, both majorize minimize iterations are derived in detail. Further information about the dataset used is also presented and a second geodesic model for subspace estimation is briefly presented and compared to the model proposed in Chapter 7.

CHAPTER 2

Background

This thesis focuses on estimating solutions to inverse problems in light-field imaging and MRI. Here, we introduce inverse problems mathematically and present a generic framework for addressing such problems. We present light-field imaging and MRI in sufficient detail for this work as two areas where inverse problems arise. Having presented concrete inverse problems, we describe the generalized Procrustes problem, and its solution which we will use in several chapters. We expound on Transform Learning, a data-driven method that has been successfully employed estimating solution for inverse problems, and which will be the basis for several of the following chapters.

2.1 Inverse Problems

When collecting samples of some signal[1] of interest, it is almost always the case that we cannot collect signal samples directly, but instead measure some function of the signal, be it blur, missing data, a projection, or something more complex. For the linear case, we can describe how the discrete measurements $\boldsymbol{y}$ are collected from the underlying (possibly continuous) true signal $\boldsymbol{x}_{\text{true}}$, with the following equation:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_{\text{true}} + \boldsymbol{\epsilon}. \tag{2.1}$$

Here, $\boldsymbol{A}$ represents the linear measurement model and $\boldsymbol{\epsilon}$ is additive white Gaussian noise (AWGN). Often $\boldsymbol{A}$ removes information from $\boldsymbol{x}$, *i.e.*, via blur or undersampling, and so $\boldsymbol{A}$ is generally wide when it can be written as a matrix. In such cases, estimating $\boldsymbol{x}_{\text{true}}$ from $\boldsymbol{y}$ and $\boldsymbol{A}$ is an underdetermined *inverse problem*, as there are many possible signals $\boldsymbol{x}$ that could give rise to the same measurements $\boldsymbol{y}$.

[1] A note on terminology: In this work, a signal is a function conveying information free of noise or artifacts. An image is a 2D signal, but the terms will occasionally be used interchangeably. Thus, we may generically refer to "image reconstruction" even when applicable to other signals, like 4D light fields.

In light of this inverse problem, several common paradigms exist for choosing an estimate $\hat{\boldsymbol{x}}$ of the true signal. Classically, linear or filtering-based methods were developed for different system models. While these methods, such as filtered back projection for CT or Wiener filtering for deconvolution problems, are fast and efficient, their results are often of insufficient quality for practical use.

Model-based image reconstruction (MBIR) is a family of nonlinear reconstruction methods that has shown to be both flexible to a variety of inverse problems, and able to produce quality estimates. As such, MBIR, as described in the following section, will be the main framework used throughout this work.

More recently, inspired by its success in computer vision, Deep Learning has been applied to a variety of inverse problems. Traditional deep learning methods require a large training dataset, and often struggle on inverse problems that are not spatially localized. Nonetheless, the signal estimates produced via these methods are of high quality, and how to best combine the benefits of both MBIR and Deep Learning is a key question of both Chapter 6 and in the greater research community.

2.1.1 Model-Based Image Reconstruction

MBIR has its roots in Bayesian maximum a posteriori (MAP) estimation. In MAP estimation, we seek to maximize the probability of our estimate given our measurements, *i.e.*, the posterior distribution $\mathsf{p}(\boldsymbol{x}|\boldsymbol{y})$. Using Bayes' Rule we can rewrite the posterior as

$$\mathsf{p}(\boldsymbol{x}|\boldsymbol{y}) = \frac{\mathsf{p}(\boldsymbol{y}|\boldsymbol{x})\,\mathsf{p}(\boldsymbol{x})}{\mathsf{p}(\boldsymbol{y})}. \tag{2.2}$$

Thus, to maximize the left-hand side, we equivalently maximize the right-hand side. When maximizing w.r.t. $\boldsymbol{x}$, we can drop $\mathsf{p}(\boldsymbol{y})$ as a constant scaling, and apply $-\log(\cdot)$ to write the problem as

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} -\log(\mathsf{p}(\boldsymbol{y}|\boldsymbol{x})) - \log(\mathsf{p}(\boldsymbol{x}))\,. \tag{2.3}$$

In the context of MAP estimation, the first term is often called the negative log-likelihood and encodes the measurement dependence of our estimate. In the case of AWGN and measurement model (2.1), the negative log-likelihood is proportional to $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2$. The second term, referred to as the negative log-prior, or just simply the prior, encodes our preconceived assumptions about which signals $\boldsymbol{x}$ are more probable irrespective of measurements. Thus, MAP provides a modular estimation framework modeling both measurement dependence and signal characteristics. Unfortunately, common signal characteristics are not easily encoded as a prior probability distribution.

MBIR relaxes the probabilistic interpretation of (2.3). To delineate from the MAP interpretation, we refer to the first term as the *data-fidelity* term instead of the negative log-likelihood, and the second term as *regularization*, instead of a prior. These MAP terms are often used informally though, despite the lack of statistical interpretability of many regularization functions.

Thus, in MBIR a common estimation problem for (2.1) may be posed as

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta\, \mathsf{R}(\boldsymbol{x}), \tag{2.4}$$

where $\mathsf{R}(\boldsymbol{x})$ is the regularization function penalizing deviation from our signal model, and β is a hyperparameter representing a trade-off between fit to data and R.

As a concrete example, we could model an image as having relatively few edges with the anisotropic TV regularizer, $\mathsf{R}(\boldsymbol{x}) = \|\boldsymbol{T}\boldsymbol{x}\|_1$, where $\boldsymbol{T}$ is the 2D finite difference operator. An estimate could then be obtained via a nonsmooth optimization algorithm, such as ADMM [13]. This regularizer does not correspond to any prior distribution, but is still useful for reconstruction.

Developing effective signal and image models and corresponding regularization function is challenging. Ideally, an image model is both broad enough to describe all plausible true images, while discriminating enough to reject noise and artifacts. Developing effective regularization is a central goal of this work.

2.1.2 Deep Learning for Inverse Problems

Instead of directly estimating a latent signal, another option is to estimate an approximate (left) inverse mapping for the forward model on the set of possible signals. This inverse mapping can then be applied to measurements to recover the latent true signal. This is often done in a supervised framework, where we assume to have a large set of clean signals $\boldsymbol{x}_i$ paired with their measurements $\boldsymbol{y}_i = \boldsymbol{A}\boldsymbol{x}_i + \boldsymbol{\epsilon}$. Given such a dataset, $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^K$, we can estimate our inverse mapping

$$\hat{f}_{\boldsymbol{\theta}}(\cdot) = f\left(\cdot\,; \hat{\boldsymbol{\theta}}\right) \text{ s.t. } \hat{\boldsymbol{\theta}} \in \arg\min_{\boldsymbol{\theta}} \sum_{i}^{K} \mathcal{L}(\boldsymbol{x}_i, f(\boldsymbol{y}_i; \boldsymbol{\theta})), \tag{2.5}$$

where f denotes a class, or an *architecture*, of functions parameterized by *weights* $\boldsymbol{\theta}$. The loss function $\mathcal{L}$ provides a metric of the quality of an estimated signal $\hat{\boldsymbol{x}}_i = f(\boldsymbol{y}_i; \boldsymbol{\theta})$ compared to the true signal $\boldsymbol{x}_i$. Generally (2.5) is minimized via some variant of stochastic (sub)gradient descent.

In artificial *neural networks*, the architecture f is designed as a composition of simpler functions, or *layers*,

$$f(\cdot\,; [\boldsymbol{\theta}_L, \boldsymbol{\theta}_{L-1}, \dots \boldsymbol{\theta}_1]) = f_L(\cdot\,; \boldsymbol{\theta}_L) \circ f_{L-1}(\cdot\,; \boldsymbol{\theta}_{L-1}) \dots \circ f_1(\cdot\,; \boldsymbol{\theta}_1), \tag{2.6}$$

where a typical layer may be $f_\ell(\boldsymbol{y}_{\ell-1}, [\boldsymbol{W}_\ell, \boldsymbol{b}_\ell]) = \sigma_\ell(\boldsymbol{W}_\ell \boldsymbol{y}_{\ell-1} + \boldsymbol{b}_\ell)$ with nonlinear *activation* function $\sigma_\ell(\cdot)$. This layered structure allows for the efficient computation of parameter gradients by applying the chain rule and backpropagating the results to earlier layers.

A major advantage of estimating an inverse mapping $\hat{f}_{\boldsymbol{\theta}}$ over iteratively reconstructing the signal estimate $\hat{\boldsymbol{x}}$ is speed. While learning $\hat{f}_{\boldsymbol{\theta}}$ can take days on modern hardware, once learned, a forward pass $\hat{\boldsymbol{x}} = \hat{f}_{\boldsymbol{\theta}}(\boldsymbol{y})$ generally takes only a fraction of the time needed for iterative reconstruction. Another advantage is that the difficult task of image modeling is learned automatically and implicitly by the network from the dataset as it learns the mapping. These networks often provide state-of-the-art performance on well-represented data and well-behaved problems.

Despite this success, modern deep learning methods have several disadvantages. First, the forward model $\boldsymbol{A}$ is often only used implicitly in the dataset. Any change to the sampling pattern or other structure of $\boldsymbol{A}$ may require re-learning the inverse mapping $\hat{f}_{\boldsymbol{\theta}}$. In addition, if a discrete model is used to retrospectively create the training pairs, modeling error can leave the learned network unusable on real measurement data. In response, several methods have included physical system updates into the network architecture to alleviate some of these challenges [68, 70, 36].

Another weakness is a tendency to hallucinate realistic false structure in images [80]. Often these networks are ill-conditioned; a small imperceptible perturbation on the input can cause noticeable structural changes on the output [32]. These problems are compounded by the lack of large and truly representative datasets for computational imaging problems. Often we collect images to look for something *exceptional*, be it cancer, product defects, or unknown developments. It is difficult to collect and well represent the exceptional. Recently, Nataraj and Otazo [58] trained over 40 state-of-the-art Deep Learning architectures on MRI brain scans with a variety of pathologies. In all cases, they found Deep Learning, with no physical modeling, to be inadequate alone at preserving unseen pathologies, even at modest undersampling levels. In some cases, networks removed large rare tumors from the images.

In light of these strengths and weakness, a natural question arises of how we can combine the flexibility and robustness of MBIR, with the speed and superior image modeling

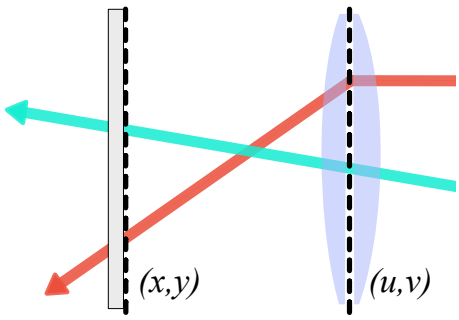


FIG 2.1 – Parameterization of all light rays within a camera using the sensor and aperture planes. All rays of interest must intersect these two planes.

of deep networks. This question will be further addressed in Chapter 6, where we consider adapting deep networks as regularization.

2.2 Light-Field Imaging

In optical imaging, it is often sufficient to characterize light from a geometric-optics perspective that treats all light as rays. If one can characterize all the rays of light within a space, then one can simulate all possible images taken within that space, and easily perform related operations, such as depth map estimation and extended depth-of-field imaging. A ray $r = (x, y, z, \theta, \phi, \lambda, t)$ is parameterized by its spatial position, its angular orientation, and its spectral wavelength, as a function of time. We would like to know the value of the *plenoptic function* that assigns a non-negative scalar intensity per volume $P(r) = P(x, y, z, \theta, \phi, \lambda, t)$ for every ray in ray space.

Characterizing the plenoptic function over an arbitrary space is difficult and rarely undertaken in practice. To simplify, often one considers only the rays of a certain wavelength in a space bounded by two planes that is free of occluders or light-sources, where light propagates freely in one general direction (see **FIG** 2.1). In this context, one can reparameterize the 5D spatio-angular coordinates of the plenoptic function in 4 dimensions: the (u, v) coordinate where rays intercept the entry plane, and the (x, y) coordinate where they intercept the exit plane. A scalar function over these free space parameterizations is called a *light field* $L(x, y, u, v)$, and one generally drops the spectral and temporal dimensions when not needed.

While this context may seem restrictive at first, it is exactly the situation that arises for light rays inside a camera. Every ray of interest in a camera must enter the camera

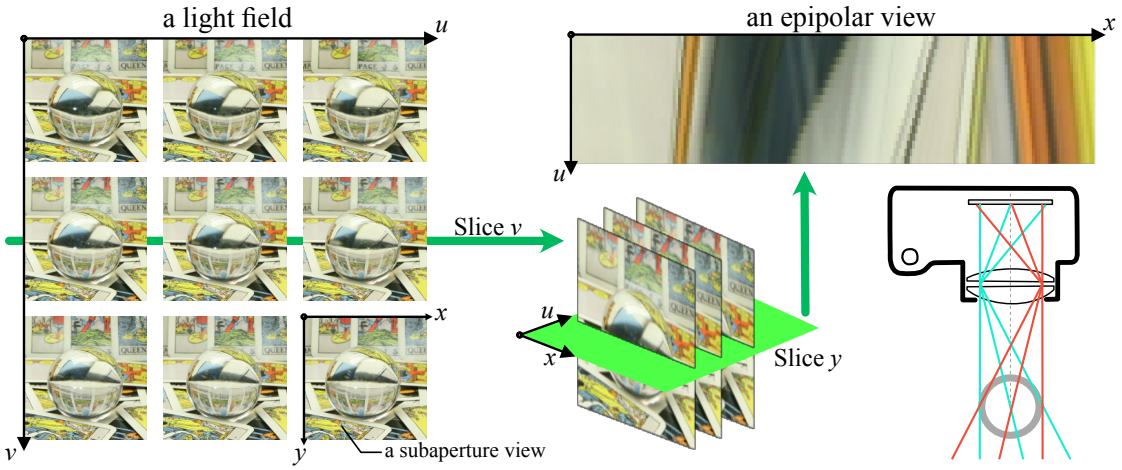


FIG 2.2 – The anatomy of a light field. (Left) An intuitive interpretation of light field is a matrix of subaperture images, where each subaperture view has a slight difference in perspective. (Center) An epipolar view is constructed by slicing one angular/perspective dimension and one spatial dimension. Here we show slicing the vertical angular coordinate v to obtain a stack of subaperture views with horizontal perspective shifts. Slicing the y dimension of this stack results in an epipolar view. (Top) In an epipolar image, nonspecular or *Lambertian* points in the scene, that emit the same ray information in all directions, draw out lines as they shift through the perspective dimension. (Bottom Right) A top-down view of the scene demonstrates why different angular coordinates have slight differences in perspective. Crystal ball light field taken from Stanford Light field Archive [79]

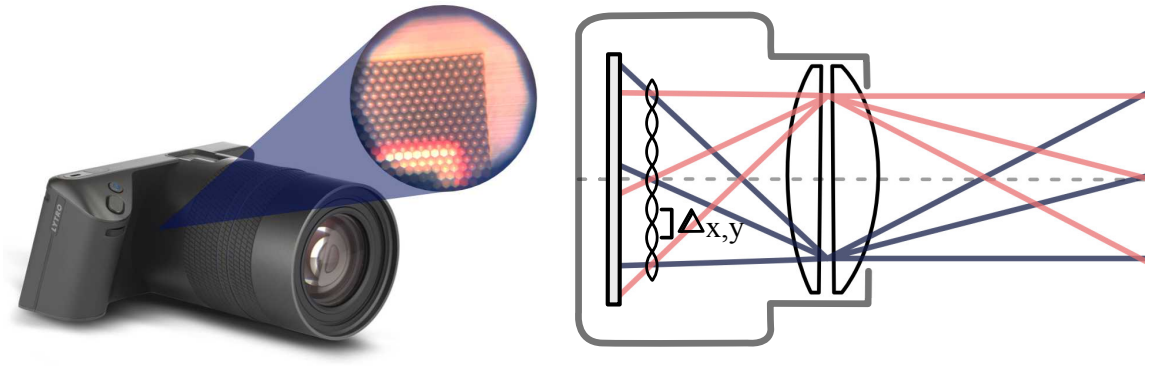


FIG 2.3 – (Left) The Lytro Illum was a handheld light field camera based on microlens technology. (Right) A diagram of how microlens cameras work. Each microlens collects light from different spatial locations and distributes it to different photosites on the sensor. This means that microlens pitch, not pixel pitch, gives us our spatial resolution.

through the aperture plane and terminate at the sensor plane. These two planes provide a natural parameterization for the rays in the camera. A light field, once acquired, can be used to simulate different focal settings by accumulating rays where they would have terminated.

An intuitive interpretation of a light field is as a 2D array of images, each with a slight shift in perspective. Each of these *subaperture images* (SAI) provides a view of the scene through a specific point in the real or simulated aperture. If instead of fixing both angular coordinates, we fix one spatial coordinate and one angular coordinate, we get what is called an *epipolar image* (EPI). **FIG** 2.2 shows an example light field in terms of both its SAI slices and an EPI slice.

2.2.1 Compressive-Light Field Photography

Handheld light field cameras, such as those made by Lytro and Raytrix, acquire the 4D light field by multiplexing angular coordinates with spatial coordinates using a microlens array as in **FIG** 2.3. In effect, each microlens acts as a miniature camera that takes a picture of the aperture plane from within the camera, so unique rays are determined by which microlens picture they end up in and where in said picture they terminate. For a fixed sensor size, this configuration reduces the measured spatial resolution by a factor of the angular resolution, leading to an undesirable trade-off.

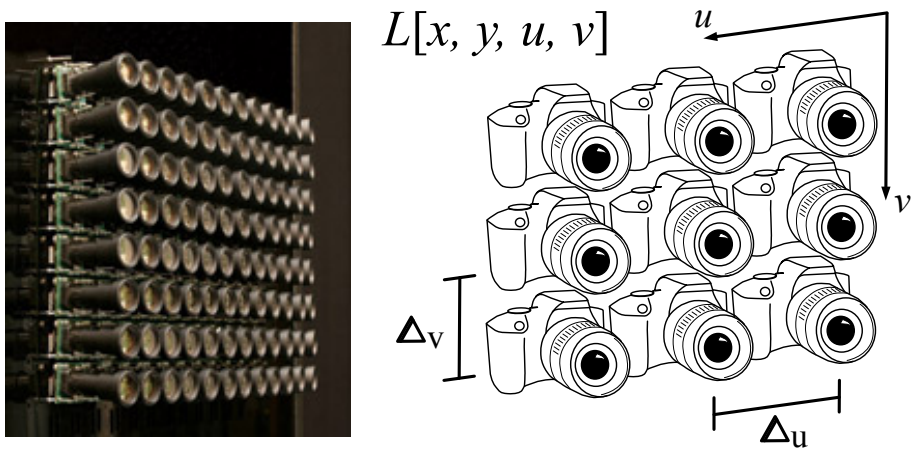


FIG 2.4 – (Left) The Stanford camera array collected many view points by positioning cameras in a grid [88]. (Right) A diagram of a camera array. The spacing between cameras controls the angular sampling rate.

Another way to capture a light field is through a camera array or camera gantry [88]. In this setup, a camera is placed at different locations along a virtual aperture plane as in **FIG** 2.4. Images acquired at each location represent the (x, y) coordinates for some fixed (u, v). For a fixed array size, camera arrays are limited in their aperture plane resolution by the physical size of the camera. Increasing the array size adds both bulk and expense. Camera gantries suffer from poor temporal resolution, due to the requirement to physically move the camera.

Despite the redundant structure of these light field dimensions, traditional light-field imaging methods are burdened with capturing the full 4D light field structure directly. Microlens-based light field cameras must trade off spatial resolution, and camera arrays must add additional bulk and expense. In response to this sampling burden and the apparent redundancy in the light field between SAI views, several compressive light-field imaging methods have been proposed.

One method is focal stack reconstruction where the light field is recovered from a series of images captured with different focal settings. While alleviating some of the sampling burden, reconstructing a light field from a focal stack presents an additional challenge: information about the light field is invariably lost due to the dimensionality gap [50]. The full 4D light field cannot be directly recovered from a 1D set of 2D measurements without enforcing additional assumptions. The forward operator for computing focal stacks

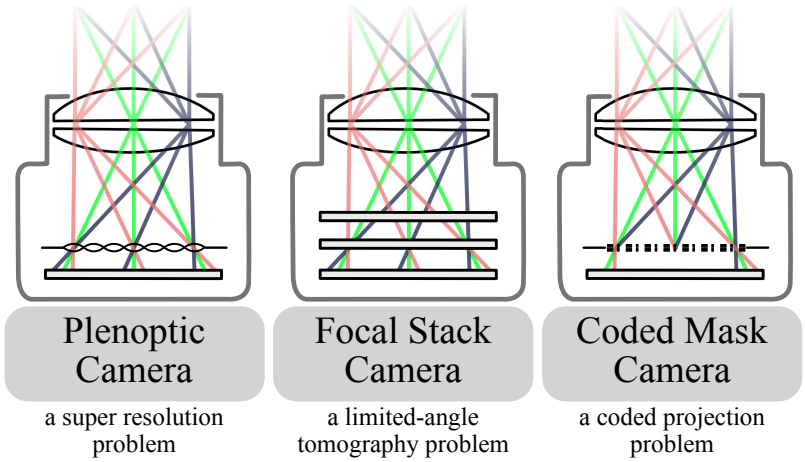


FIG 2.5 – A few cameras for compressive light-field capture. The plenoptic camera directly samples a light-field by sacrificing spatial resolution. Focal-stack cameras and coded-mask cameras collect 2D measurements and must reconstructed the 4D light field.

from light fields consists of a series of parallel plane integrals, and shares many similarities with the Radon transform used in CT reconstruction. Appendix A derives a discretized forward model for focal stack reconstruction.

Other compressive light field cameras include coded mask and coded aperture cameras, which place a known transmission mask near the sensor or aperture, respectively, to encode angular variation of rays within a camera [53]. **FIG** 2.5 depicts several of the light-field cameras discussed here.

2.3 Magnetic Resonance Imaging

A full exposition of the physics of MRI is unnecessary for the purposes of this work. Here, we briefly review the details of single-coil imaging for MRI. In single-coil imaging, measurements $\boldsymbol{y}$ are collected sequentially in the spatial Fourier space, *i.e.*, *k-space*, of the true image. When k-space is fully sampled, the image can be reconstructed using an inverse DFT. To accelerate scan times, k-space is often undersampled and standard reconstruction methods result in spatial aliasing artifacts.

In the context of MRI, our system model $\boldsymbol{A}$ can be written as $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{F}$ where $\boldsymbol{F}$ is a (Discrete) Fourier Transform and $\boldsymbol{U}$ is an undersampling operator. This forward model has the nice property that the hessian of the standard data-fidelity term $\boldsymbol{A}'\boldsymbol{A}$ is efficiently diagonalizable by the DFT. When the data update involves minimizing a quadratic cost function, this will often allow for a closed-form update that is implementable on large-scale problems.

2.4 Generalized Procrustes Problems

A key result we will leverage throughout this thesis to efficiently update adaptive regularizers is the solution to a generalized Procrustes problem [71]. We let $\mathcal{V}^{N\times k}(\mathbb{F})$ denote the Stiefel Manifold

$$\mathcal{V}^{N\times k}(\mathbb{F}) = \left\{ \boldsymbol{Q} \in \mathbb{F}^{N\times k} : \boldsymbol{Q}'\boldsymbol{Q} = \boldsymbol{I} \right\}, \tag{2.7}$$

i.e., the set of $N \times k$ matrices with orthonormal columns. The field $\mathbb{F}$ may be the real numbers $\mathbb{R}$ or the complex numbers $\mathbb{C}$ and we will drop it from the notation when the result is generic over the choices or it is implied by the problem.

The generalized Procrustes problems is then given by

$$\begin{aligned}
\hat{\boldsymbol{Q}} &= \underset{\boldsymbol{Q}\in\mathcal{V}^{N\times k}}{\arg\min} \left\| \boldsymbol{Q}\boldsymbol{A} - \boldsymbol{B} \right\|_{\mathrm{F}}^2 && (2.8)\\
&= \underset{\boldsymbol{Q}\in\mathcal{V}^{N\times k}}{\arg\min} \left\| \boldsymbol{A} \right\|_{\mathrm{F}}^2 - 2\,\mathrm{real}\{\mathsf{trace}\{\boldsymbol{Q}'\boldsymbol{B}\boldsymbol{A}'\}\} + \left\| \boldsymbol{B} \right\|_{\mathrm{F}}^2 && (2.9)\\
&= \underset{\boldsymbol{Q}\in\mathcal{V}^{N\times k}}{\arg\max}\, \mathrm{real}\{\mathsf{trace}\{\boldsymbol{Q}'\boldsymbol{B}\boldsymbol{A}'\}\}, && (2.10)
\end{aligned}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are given matrices of the appropriate size. The solution is given by the SVD of $\boldsymbol{B}\boldsymbol{A}'$

$$\begin{aligned}
\hat{\boldsymbol{Q}} &= \boldsymbol{U}\boldsymbol{V}' && (2.11)\\
&\quad \text{where } \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}' \text{ is the (thin) SVD of } \boldsymbol{B}\boldsymbol{A}'. && (2.12)
\end{aligned}$$

While computing an SVD on large problems can be computationally intensive, we will apply this result on smaller image patches where the SVD can be computed efficiently.

2.5 Analysis Sparsity Models

Broadly speaking, sparsity signal models for inverse problems can be categorized as being *synthesis* type or *analysis* type. In a linear synthesis setting, we could model a true image $\boldsymbol{x}_{\mathrm{true}}$ as $\boldsymbol{x}_{\mathrm{true}} \approx \boldsymbol{D}\boldsymbol{z}$ where $\boldsymbol{D}$ is often referred to as a dictionary or (overcomplete) basis and $\boldsymbol{z}$ is sparse. Thus, we say $\boldsymbol{x}_{\mathrm{true}}$ is (approximately) synthesized from the columns of $\boldsymbol{D}$. In a linear analysis setting, we model $\boldsymbol{x}_{\mathrm{true}}$ in some transformed space, *i.e.*, we say $\boldsymbol{T}\boldsymbol{x}_{\mathrm{true}} \approx \boldsymbol{z}$ where $\boldsymbol{z}$ is sparse. $\boldsymbol{T}$ is referred to as a transform or analysis operator. Anisotropic TV is an example of regularization employing a (strict) analysis sparsity model, $\mathsf{R}(\boldsymbol{x}) = \left\| \boldsymbol{T}\boldsymbol{x} \right\|_1$, where $\boldsymbol{T}$ is the finite difference operator.

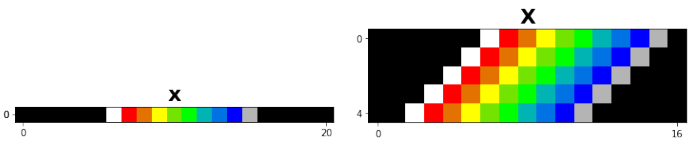


FIG 2.6 – Extracting 1D patches of length 5 from the 1D vector $\boldsymbol{x}$ yields the patch matrix $\boldsymbol{X}$ on the right. $\boldsymbol{X}^T$ is the correlation matrix of $\boldsymbol{x}$

While similar, these different models are generally distinct. If the approximation is taken to be a strict equality, then they are equivalent when $\boldsymbol{D} = \boldsymbol{T}^{-1}$, while if the approximation is interpreted as having a small ℓ_2 difference, then they are equivalent when the (stricter) unitary condition $\boldsymbol{D}^{-1} = \boldsymbol{D}' = \boldsymbol{T}$ holds. Many analysis and synthesis models are not square though and so cannot be considered equivalent in either sense.

Often, due to data size and image self-similarities, it is easier to model the local — instead of global — properties of an image. In other words, we assume $\boldsymbol{T}\boldsymbol{P}_j\boldsymbol{x}$ is approximately sparse, where $\boldsymbol{P}_j$ is an $n \times N$ matrix of 0 and 1 elements that extracts the jth, patch or window of n elements from the image, and $\boldsymbol{T}$ is a transform that sparsifies the patch. A patch-wise or local transform sparsity regularization could then be written as

$$\mathsf{R}(\boldsymbol{x}) = \min_{\{\boldsymbol{z}_j\}} \sum_j \|\boldsymbol{T}\boldsymbol{P}_j\boldsymbol{x} - \boldsymbol{z}_j\|_2^2 + \gamma\|\boldsymbol{z}_j\|_p \tag{2.13}$$

where p is chosen to encourage sparsity in $\boldsymbol{z}_j$, typically $p \in [0, 1]$.

For notational convenience, we often let $\boldsymbol{X} = [\boldsymbol{P}_1\boldsymbol{x} \dots \boldsymbol{P}_J\boldsymbol{x}]$ and similarly $\boldsymbol{Z} = [\boldsymbol{z}_1 \dots \boldsymbol{z}_J]$. We can then simplify (2.13) as[2]

$$\mathsf{R}(\boldsymbol{x}) = \min_{\boldsymbol{Z}} \|\boldsymbol{T}\boldsymbol{X} - \boldsymbol{Z}\|_{\mathrm{F}}^2 + \gamma\|\mathsf{vec}(\boldsymbol{Z})\|_p. \tag{2.14}$$

Interestingly, when the set of $\boldsymbol{P}_j$'s extracts maximally overlapping patches, each row of $\boldsymbol{T}$ in (2.13) performs an inner product with a sliding window in $\boldsymbol{x}$, which is equivalent to filtering via correlation [62, 68]. Put another way, $\boldsymbol{X}^T$ is the correlation matrix of image $\boldsymbol{x}$. **FIG** 2.6 provides an illustration of a simple vector $\boldsymbol{x}$ and patch matrix $\boldsymbol{X}$. Thus, we can

[2]Because our images are often natively 2D, $\|\boldsymbol{x}\|_2$ can sometimes be ambiguous as to whether it refers to a vector or matrix 2-norm. In this work, we denote all matrix norms with $\|\cdot\|$ and all vector norms with $\|\cdot\|$ to remove any ambiguity. Of course, there is no ambiguity here for the Frobenius matrix norm.

rewrite (2.13) equivalently as

$$\mathsf{R}(\boldsymbol{x}) = \min_{\{\boldsymbol{Z}_{i,:}\}} \sum_{i}^{n} \left\| \boldsymbol{T}^{*}_{i,:} \star \boldsymbol{x} - \boldsymbol{Z}_{i,:} \right\|_2^2 + \gamma \| \boldsymbol{Z}_{i,:} \|_p \tag{2.15}$$

where $\boldsymbol{Z}_{i,:}$ are sparse images, $\boldsymbol{T}_{i,:}$ is the ith row of $\boldsymbol{T}$, and the filtering is performed in the natural dimensions of $\boldsymbol{x}$ (2D for grayscale images). For this reason, we sometimes refer to a patch-wise transform $\boldsymbol{T}$ as a filter bank and to the rows of $\boldsymbol{T}$ as filters. In this work, we will prefer the matrix product notation of (2.13), but which is preferred for implementation will depend on other factors.

2.5.1 Population-Adaptive Transform Learning

Classically, transforms — such as finite differences, the DCT, wavelets, curvelets, platelets, *etc.*— have been derived to be optimal over some class of images. Often these classes are restrictive for mathematical necessity, such as a class of piecewise constant images. An arguably more appropriate, but less algebraic, choice of transform is one that minimizes the expected value of the regularizer (2.13) over the distribution of possible true images

$$\bar{\boldsymbol{T}} = \underset{\boldsymbol{T} \in \mathcal{T}}{\arg\min}\, \mathbb{E}_{\mathrm{x}} \left[\min_{\{\boldsymbol{z}_j\}} \sum_{j} \| \boldsymbol{T} \boldsymbol{P}_j \boldsymbol{x} - \boldsymbol{z}_j \|_2^2 + \gamma \| \boldsymbol{z}_j \|_p \right]. \tag{2.16}$$

The set $\mathcal{T}$ represents the set of feasible $\boldsymbol{T}$, and should, at a minimum, exclude the trivial solution $\boldsymbol{T} = \boldsymbol{0}$.

We cannot describe the distribution of true images in closed-form, but given a (large) set of K training signals $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K\}$ we can estimate[3] (2.16) by minimizing the following cost function

$$\hat{\boldsymbol{T}} = \underset{\boldsymbol{T} \in \mathcal{T}}{\arg\min}\, \frac{1}{K} \sum_{k=1}^{K} \left(\min_{\{\boldsymbol{z}_{j,k}\}} \sum_{j} \| \boldsymbol{T} \boldsymbol{P}_j \boldsymbol{x}_k - \boldsymbol{z}_{j,k} \|_2^2 + \gamma \| \boldsymbol{z}_{j,k} \|_p \right) \tag{2.17}$$

$$= \underset{\boldsymbol{T} \in \mathcal{T}}{\arg\min} \min_{\{\boldsymbol{Z}_k\}} \frac{1}{K} \left(\| \boldsymbol{T} [\boldsymbol{X}_1 \ldots \boldsymbol{X}_K] - [\boldsymbol{Z}_1 \ldots \boldsymbol{Z}_K] \|_{\mathrm{F}}^2 + \gamma \| \mathsf{vec}([\boldsymbol{Z}_1 \ldots \boldsymbol{Z}_K]) \|_p \right). \tag{2.18}$$

A common choice for $\mathcal{T}$ is the set of unitary matrices, which we denote $\mathcal{V}^{n \times n}$. While restrictive, this choice emits a closed-form solution for (2.18) with fixed $\boldsymbol{Z}$ as a Procrustes

[3] A simple sufficient, but not necessary, condition is for the training samples to be jointly independent, but not necessarily pairwise independent.

problem. Other classes of transforms, in conjunction with penalty functions, have been proposed, such as linear independent columns [67], or Fourier incoherence of rows [62].

Once a estimate $\hat{\boldsymbol{T}}$ has been obtained, it can be applied to reconstruct an image by minimizing cost (2.4) with regularizer (2.13)

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \left(\min_{\{\boldsymbol{z}_j\}} \sum_j \left\| \hat{\boldsymbol{T}} \boldsymbol{P}_j \boldsymbol{x} - \boldsymbol{z}_j \right\|_2^2 + \gamma \|\boldsymbol{z}_j\|_p \right). \tag{2.19}$$

A common approach to minimizing (2.19) is block coordinate descent (BCD), which alternates between a simple quadratic $\boldsymbol{x}$ update and evaluating $\operatorname{prox}_{\ell_p}\left(\boldsymbol{T}\boldsymbol{P}_j\boldsymbol{x}, \frac{\gamma}{2}\right)$ for each $\boldsymbol{z}_j$.

Transform learning methods have been applied in the context of 2D image denoising [66], MR image reconstruction from undersampled k-space measurements [67], and video denoising [87].

2.5.2 Instance-Adaptive (Blind) Transform Learning

Instead of minimizing the expected value of the regularizer (2.13) over a population of images as described previously, we could minimize (2.13) with respect to a particular instance. Ideally, we would minimize with respect to the image we are reconstructing $\boldsymbol{x}_{\text{true}}$, but, as this is not available, we can approximate it with our current best estimate $\hat{\boldsymbol{x}}$ each iteration of the reconstruction. Thus, we estimate $\boldsymbol{T}$ *blindly*, *i.e.*, without knowledge of a true $\boldsymbol{x}$, by using a regularizer of the following form that has an internal minimization problem within it:

$$\mathsf{R}(\boldsymbol{x}) = \min_{\boldsymbol{T}\in\mathcal{T}} \min_{\{\boldsymbol{z}_j\}} \sum_j \|\boldsymbol{T}\boldsymbol{P}_j\boldsymbol{x} - \boldsymbol{z}_j\|_2^2 + \gamma \|\boldsymbol{z}_j\|_p. \tag{2.20}$$

We generally minimize the resulting cost function using BCD like described previously, only now $\boldsymbol{T}$ is updated every iteration.

Despite their clear structural similarity, population- and instance-adaptive transform learning represent different assumptions about our class of true images. A population-specific transform is more constrained, but the learned filters represent features of the population. An instance-specific transform is more flexible, but only enforces that the image estimate is sparsifiable, not sparse under any specific transform. Chapter 5 further investigates when one choice of model may be preferable over the other.

2.5.2.1 Blind Unitary Transform Learning

Common choices for applying blind transform learning are $\mathcal{T} = \mathcal{V}^{n\times n}$, the set of all unitary matrices of size $n \times n$, and $p = 0$. Here, we describe the common approach for minimizing a regularizer with these parameters.

We apply blind unitary transform learning as a regularizer for the problem of recovering an image $\boldsymbol{x}$ from measurements $\boldsymbol{y}$ by minimizing the following cost function using block coordinate descent (BCD):

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \min_{\{\boldsymbol{z}_j\}} \min_{\boldsymbol{T}\in\mathcal{V}^{n\times n}} \lambda\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \sum_j \|\boldsymbol{T}\boldsymbol{P}_j\boldsymbol{x} - \boldsymbol{z}_j\|_2^2 + \gamma^2\|\boldsymbol{z}_j\|_0. \tag{2.21}$$

Again, $\boldsymbol{A} \in \mathbb{R}^{M\times N}$ represents our system model that generated vectorized measurements $\boldsymbol{y} \in \mathbb{R}^M$. Each matrix $\boldsymbol{P}_j \in \{0,1\}^{n\times N}$ extracts the jth patch of n elements from a vectorized image $\boldsymbol{x}$ and we sum over all such j with overlapping windows of stride 1. The shape of these patches, or windows, is a hyperparameter of this model and must be chosen beforehand. Generally, we want the patches to be large enough to capture meaningful features in each dimensions. Here, $\|\cdot\|_0$ denotes the so-called zero "norm" or counting measure (number of nonzero vector elements).

While (2.21) is a non-convex cost function, due both to the non-convex zero "norm" and product between $\boldsymbol{T}$ and $\boldsymbol{x}$, applying BCD to it is globally convergent, *i.e.*, BCD converges to a local minimizer from any initial starting point [67].

A BCD method alternates between minimizing the sparse codes $\{\boldsymbol{z}_j\}$, the unitary transform $\boldsymbol{T}$, and the image $\boldsymbol{x}$. Minimizing (2.21) with respect to $\boldsymbol{z}_j$ leads to the following proximal problem with known closed-form solution:

$$\hat{\boldsymbol{z}}_j = \arg\min_{\boldsymbol{z}_j} \|\boldsymbol{T}\boldsymbol{P}_j\boldsymbol{x} - \boldsymbol{z}_j\|_2^2 + \gamma^2\|\boldsymbol{z}_j\|_0 \tag{2.22}$$

$$= \mathsf{hard}_\gamma(\boldsymbol{T}\boldsymbol{P}_j\boldsymbol{x}) \tag{2.23}$$

where $\mathsf{hard}_\gamma(\cdot)$ is element-wise hard-thresholding by threshold γ

$$\mathsf{hard}_\gamma(a) = \begin{cases} 0 & |a| \le \gamma \\ a & |a| > \gamma. \end{cases} \tag{2.24}$$

Compared to dictionary learning, where ℓ_0-sparse coding is NP-Hard and requires an expensive step such as Orthogonal Matching Pursuit (OMP), transform learning provides a simple closed-form sparse-code update.

Algorithm 1 Blind UTL

Require: $\boldsymbol{x}^{(0)}, \boldsymbol{T}^{(0)}$ $\boldsymbol{y}$, $\boldsymbol{A}$, $\lambda, \gamma > 0$
  Let $\boldsymbol{W} = \sum_j \boldsymbol{P}_j' \boldsymbol{P}_j$
  for $i = 1, \ldots, I$ **do**
    Construct $\boldsymbol{X} = [\boldsymbol{P}_1 \boldsymbol{x}^{(i-1)} \ldots \boldsymbol{P}_j \boldsymbol{x}^{(i-1)} \ldots \boldsymbol{P}_J \boldsymbol{x}^{(i-1)}]$
    $\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V}' = \text{SVD}\big(\boldsymbol{Z}^{(i-1)} \boldsymbol{X}'\big)$
    $\boldsymbol{T}^{(i)} = \boldsymbol{U}\boldsymbol{V}'$
    $\boldsymbol{Z}^{(i)} = \mathsf{hard}_\gamma\big(\boldsymbol{T}^{(i)} \boldsymbol{X}\big)$
    Construct $\bar{\boldsymbol{x}} = \sum_j \boldsymbol{P}_j' \boldsymbol{T}'^{(i)} \boldsymbol{Z}_{:,j}^{(i)}$
    $\boldsymbol{x}^{(i)} = (\lambda \boldsymbol{A}'\boldsymbol{A} + \boldsymbol{W})^{-1} (\lambda \boldsymbol{A}'\boldsymbol{y} + \bar{\boldsymbol{x}})$
  end for

Defining $\boldsymbol{X} = [\boldsymbol{P}_1 \boldsymbol{x} \ldots \boldsymbol{P}_J \boldsymbol{x}]$ and $\boldsymbol{Z} = [\boldsymbol{z}_1 \ldots \boldsymbol{z}_J]$, we rewrite the $\boldsymbol{T}$ update as a Procrustes problem with known closed-form solution:

$$\begin{aligned} \hat{\boldsymbol{T}} &= \underset{\boldsymbol{T} \in \mathcal{V}^{n \times n}}{\arg\min} \sum_j \|\boldsymbol{T}\boldsymbol{P}_j \boldsymbol{x} - \boldsymbol{z}_j\|_2^2 \\ &= \underset{\boldsymbol{T} \in \mathcal{V}^{n \times n}}{\arg\min} \|\!|\boldsymbol{T}\boldsymbol{X} - \boldsymbol{Z}|\!\|_{\mathrm{F}}^2 = \boldsymbol{U}\boldsymbol{V}' \end{aligned} \tag{2.25}$$

where $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, $\boldsymbol{V}'$ denotes the SVD of $\boldsymbol{Z}\boldsymbol{X}'$. Because $\boldsymbol{T}$ is an $n \times n$ matrix, where n is the number of elements in a patch, this SVD is performed on a relatively small matrix.

The image update is then a standard quadratic minimization problem

$$\begin{aligned} \hat{\boldsymbol{x}} &= \underset{x}{\arg\min}\, \lambda \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \sum_j \|\boldsymbol{T}\boldsymbol{P}_j \boldsymbol{x} - \boldsymbol{z}_j\|_2^2 \\ &= (\lambda \boldsymbol{A}'\boldsymbol{A} + \sum_j \boldsymbol{P}_j' \boldsymbol{P}_j)^{-1} (\lambda \boldsymbol{A}'\boldsymbol{y} + \sum_j \boldsymbol{P}_j' \boldsymbol{T}' \boldsymbol{z}_j). \end{aligned} \tag{2.26}$$

Because $\sum_j \boldsymbol{P}_j' \boldsymbol{P}_j$ is a diagonal matrix, system models that have an efficiently diagonalizable Hessian matrix $\boldsymbol{A}'\boldsymbol{A}$, such as denoising, inpainting or deblurring, are efficiently computed in closed-form. For all other cases, running a few iterations of conjugate gradient will still descend the cost and converge on a solution.

Algorithm 1 summarizes a patch-wise implementation of blind unitary transform learning as described above. As described before in (2.15), a filtering interpretation can be understood by examining the relationship of the rows of $\boldsymbol{T}$ with the original image $\boldsymbol{x}$. Each row of $\boldsymbol{T}$ performs an inner product with a sliding window in $\boldsymbol{x}$, which is equivalent to

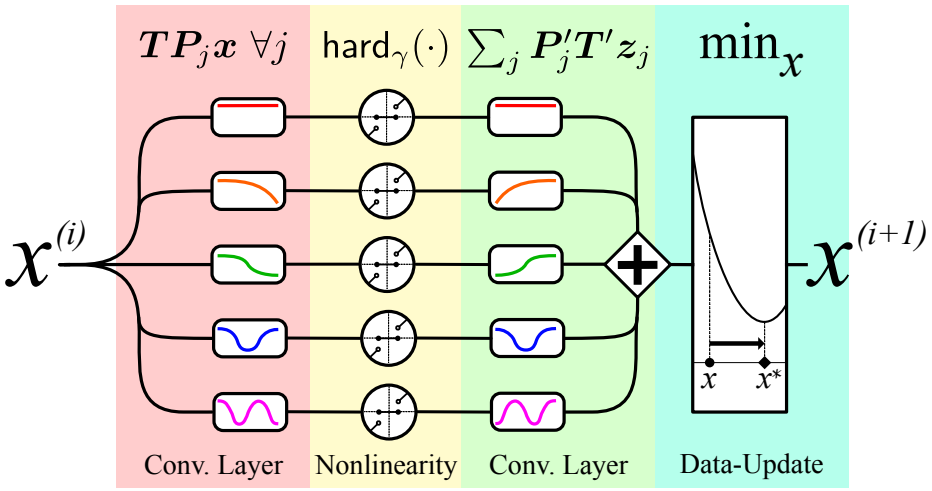


FIG 2.7 – Regularization based on transform learning can be interpreted as a filter bank followed by a data update term. A filter bank can be interpreted as a shallow Convolutional Neural Network (CNN). The red and yellow regions correspond to (2.23) and the green and blue regions correspond to (2.26). (The update of the transform $\boldsymbol{T}$ in (2.25) each iteration is not pictured.)

filtering via convolution [62]. Thus, the set of sparse codes $\boldsymbol{Z}$ represents the thresholded output of a filter bank of n filters, where n is the number of rows of $\boldsymbol{T}$. Applying the inverse transform $\boldsymbol{T}'$ and aggregating is equivalent to filtering with matched filters and summing the channels. **FIG** 2.7 shows a diagram of the flow of $\boldsymbol{x}$ in one iteration. (Note that it does not show the update of the filters $\boldsymbol{T}$). Thus, we can interpret each iteration of blind unitary transform learning as an *instance-adaptive* shallow CNN, where the filters are learned iteratively in an unsupervised fashion, followed by a data update that incorporates our prior knowledge on $\boldsymbol{y}$ and $\boldsymbol{A}$. Ravishankar *et al.* [68] builds on this CNN interpretation by unrolling the iterations of UTL as layers of a deep neural network and training the filters of each layer independently and greedily.

We note that the unitary constraint is particularly well suited for the blind or instance-adaptive transform learning approach because it is computationally efficient, and can, thus, be computed at reconstruction time, and not because it is likely to be an ideal class of efficient sparsifying transforms.[4] Empirically, it performs well and Chapter 5 further compares UTL in both the blind and non-blind setting. In the non-blind case, better options for learning sparsifying transforms likely exist, *e.g.*, the Analysis K-SVD [69]. The use of blind unitary transform learning has been shown to be an effective regularizer for

[4]Several popular classical transforms — such as the DCT or orthogonal wavelets — do belong to this class though.

MR images [67]. The next chapter investigates the use of unitary transform learning for higher-dimensional light field patches.

CHAPTER 3

Blind Unitary Transform Learning for Inverse Problems in Light-Field Imaging

Light-field cameras have enabled a new class of digital post-processing techniques. Unfortunately, the sampling requirements needed to capture a 4D color light field directly using a microlens array requires sacrificing spatial resolution and SNR in return for greater angular resolution. Because recovering the true light field from focal-stack data is an ill-posed inverse problem, we propose using blind unitary transform learning (UTL) as a regularizer. UTL attempts to learn a set of filters that maximize the sparsity of the encoded representation. This paper investigates which dimensions of a light field are most sparsifiable by UTL and lead to the best reconstruction performance. We apply the UTL regularizer to light field inpainting and focal stack reconstruction problems and find it improves performance over traditional hand-crafted regularizers.

3.1 Introduction

3.1.1 Inverse Problems

Reconstructing a light field from a set of compressed or subsampled measurements is an underdetermined inverse problem. There can be many possible light fields that will perfectly match our data. Thus, a model is needed to select one of the many candidate light fields, by choosing one that is consistent with our assumptions about the true light field's properties. A common paradigm that we use in this work is to include the model as regularization in a minimization problem

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \lambda \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \mathsf{R}(\boldsymbol{x}) \tag{3.1}$$

This chapter based on the author's published work at ICCVW 2019 [10].

where $\boldsymbol{A}$ is a wide matrix encoding the linear operation relating the unknown light field $\boldsymbol{x}$ to the measurement $\boldsymbol{y}$, λ is a hyperparameter representing our confidence in the measurements, and $\mathsf{R}(\boldsymbol{x})$ is a regularization function representing our signal model.

A number of previous works attempt to restore a light field from a set of compressed or corrupted measurements, such as view inpainting, focal stack reconstruction, coded aperture reconstruction, super-resolution, denoising, and inpainting. A majority of these approaches can be divided into linear filtering based methods [21, 22, 50], depth-estimation-dependent methods [51, 55, 48], deep learning methods [27, 42, 56, 89, 91, 95], and low-rank or sparse methods [3, 4, 9, 23, 25, 40, 41, 53, 54, 73, 77, 83, 84]. Most of the sparsity based methods assume a hand-crafted transform, such as the discrete cosine transform (DCT) [54] or shearlets [83, 84]. A notable exception is [53] that applies K-SVD to learn a dictionary for light field patches from training data *a priori*. While using hand-crafted transforms, LF-BM5D [3] does employ instance-adaptive thresholding and filtering.

Transform sparsity models data as being locally sparsifiable. In other words, we assume $\boldsymbol{T}\boldsymbol{P}_j\boldsymbol{x}$ is sparse, where $\boldsymbol{P}_j$ is a matrix of 0 and 1 elements that extracts the jth, for example, $p_x \times p_y \times p_u \times p_v \times p_c$ patch or window from the data, and $\boldsymbol{T}$ is a transform that sparsifies the patch. Compared to dictionary methods, that generally synthesize a signal vector from a set of sparse codes, transform sparsity encourages a signal to be sparsifiable. These conditions are not necessarily equivalent, except in the uncommon case when the dictionary and transform are inverses of each other.

In transform learning, we attempt to learn a transform from data, instead of using a hand-crafted transform such as wavelets or the DCT. There are multiple modes of transform learning. One mode is to use a set of training signals $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K\}$ and learn a transform $\boldsymbol{T}$ that is effective for sparsifying patches drawn from those signals. In words, we want $\boldsymbol{T}$ such that $\boldsymbol{T}\boldsymbol{P}_j\boldsymbol{x}_k$ is typically sparse. One way this can be done is by minimizing the following cost function

$$\hat{\boldsymbol{T}} = \underset{\boldsymbol{T}\in\mathcal{V}^{n\times n}}{\arg\min} \min_{\{\boldsymbol{z}_{j,k}\}} \sum_{k=1}^{K} \sum_{j} \|\boldsymbol{T}\boldsymbol{P}_j\boldsymbol{x}_k - \boldsymbol{z}_{j,k}\|_2^2 + \gamma^2 \|\boldsymbol{z}_{j,k}\|_0, \tag{3.2}$$

where $\mathcal{V}^{n\times n}$ denotes the set of unitary matrices. This approach bears many similarities to a standard dictionary learning formulation

$$\hat{\boldsymbol{D}} = \underset{\boldsymbol{D}}{\arg\min} \min_{\{\boldsymbol{z}_j\}} \sum_{k=1}^{K} \sum_{j} \|\boldsymbol{P}_j\boldsymbol{x}_k - \boldsymbol{D}\boldsymbol{z}_{j,k}\|_2^2 + \gamma^2 \|\boldsymbol{z}_{j,k}\|_0. \tag{3.3}$$

Transform learning methods have been applied in the context of 2D image denoising [66], MR image reconstruction from undersampled k-space measurements [67], and video denoising [87].

3.1.2 Contributions

Because of the unitary invariance of the ℓ_2 norm, unitary transform learning is equivalent to a formulation of unitary dictionary learning. Thus, our proposed method is most similar to that of Marwah *et al.* [53]. The work proposed here differs in two major aspects.

First, we do not learn our transforms from training data *a priori*. We instead opt for a *blind* UTL method that learns sparsifying transforms blindly in an instance-adaptive fashion. To the authors' knowledge, this is the first time instance-adaptive transform or dictionary sparsity has been applied to light-field imaging.

Second, we investigate the sparsifiability of different dimensions of the light field. Marwah *et al.* [53] used a 5D (4D + color) light field patch in learning and fitting their dictionary. While dictionary atoms describing epipolar patches or spatial patches could, in theory, be learned inside of a 5D patch, often dense full-dimensional patches are learned. Due to the non-convexity of these learning methods, it is not given that the learned dense 5D patches are optimal, though we hope they approximate this well. Indeed, much of the prior work can be divided among epipolar methods [83, 84, 89] and 4D+ methods [3, 40, 54, 56].

This work explores multiple approaches to choosing light field patches for transform and dictionary learning, including subaperture image (SAI) patches (x, y, c), epipolar image (EPI) patches in both the horizontal (x, u, c) and vertical (y, v, c) directions as well as full-dimensional light field (LF) patches (x, y, u, v, c). In a hand-crafted and pre-trained setting, applying a method only spatially along subaperture images completely ignores light field structure. In contrast, in the blind setting, features can in theory be learned more effectively due to the light field redundancy. As different light-field imaging applications may benefit more from different types of patches, we compare different patch dimension choices on a couple of inverse problems in light-field imaging: inpainting and reconstruction from focal stack images.

Section 3.2 provides a general description of unitary transform learning (UTL) as used in this work. For a more detailed description of UTL, including a convergence analysis, see [67]. Section 3.3 applies UTL with different patch structures to inverse problems in light-field imaging. Section 3.4 compares the performance of the different methods and analyzes the learned transforms.

3.2 Methods

We apply blind unitary transform learning as a regularizer for the problem of recovering a light field $\boldsymbol{x}$ from measurements $\boldsymbol{y}$ by minimizing the following cost function using block coordinate descent (BCD):

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in\mathbb{R}^N}{\arg\min} \underset{\{\boldsymbol{z}_j\in\mathbb{R}^N\}}{\min} \underset{\boldsymbol{T}\in\mathcal{V}^{n\times n}}{\min} \lambda\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|_2^2 + \sum_j \|\boldsymbol{T}\boldsymbol{P}_j\boldsymbol{x}-\boldsymbol{z}_j\|_2^2 + \gamma^2\|\boldsymbol{z}_j\|_0. \tag{3.4}$$

We let $\boldsymbol{A}\in\mathbb{R}^{M\times N}$ represent our system model that generated vectorized measurements $\boldsymbol{y}\in\mathbb{R}^M$. Here $\boldsymbol{P}_j\in\{0,1\}^{n\times N}$ is a matrix that extracts the jth $p_x\times p_y\times p_u\times p_v\times p_c$ patch from a vectorized light field $\boldsymbol{x}$ and we sum over all such j with overlapping windows of stride 1. Here, $\|\cdot\|_0$ denotes the so-called zero "norm" or counting measure (number of nonzero vector elements). We follow the steps outlined in Section 2.5.2.1 to minimize (3.4) using BCD for UTL.

In this work, we propose 3 variations of (3.4), namely LF UTL, EPI UTL and SAI UTL, that differ only in the shape of the patches extracted by $\boldsymbol{P}_j$. For SAI UTL, we constrain $p_u = p_v = 1$ so that we only learn spatial patches. For EPI UTL, we constrain either $p_x = p_u = 1$ or $p_y = p_v = 1$ to only learn features in the epipolar dimension. Finally, for LF UTL, we learn full-dimensional patches. In all variations, we include all 3 color dimensions in every patch ($p_c = 3$). We divide our approaches in this manner, so that we can better understand the potential of each of these dimensions.

Because $\sum_j \boldsymbol{P}_j^T\boldsymbol{P}_j$ is a diagonal matrix, system models that have a diagonalizable Hessian matrix $\boldsymbol{A}^T\boldsymbol{A}$, such as denoising, inpainting or deblurring, are efficiently computed in closed-form. For all other cases, running a few iterations of conjugate gradient provides a suitable approximation. In the case of the focal stack system model, I experimented with further accelerating conjugate gradient and other smooth optimization methods with circulant preconditioning. While the focal stack Hessian matrix is Toeplitz, the fact that my angular dimensions, u,v, were each only 5 wide resulted in edge artifacts dominating when preconditioning with a circulant matrix. Interestingly, the original Hessian matrix already has most of its energy near the main diagonal. The fact that the projections are over dimensions only 5 wide means that the operations are still highly localized, and common preconditioning choices provided little improvement.

FIG 3.1 – Central subaperture images from The (New) Stanford Light Field Archive [79]. From left to right, (Top) amethyst, Lego knights, crystal ball, bracelet, jelly beans, bunny (Bottom) eucalyptus, treasure chest, Lego bulldozer, Lego truck. Images resized independently.

3.3 Experiments

We validated the proposed method on 10 light fields from the Stanford Light Field Dataset [79]; see **FIG** 3.1. From each light field, we extracted the central 5×5 views and spatially downsampled by a factor of 3 for testing our method. For each patch shape, we tuned all hyperparameters, unless otherwise stated, using the Tree of Parzen Estimators as implemented in the `hyperopt` Python package [8]. For hyperparameter tuning, we used smaller $5 \times 5 \times 192 \times 192$ light fields cropped from the bunny, crystal ball, and Lego bulldozer light fields to reduce tuning time. We used peak signal-to-noise ratio (PSNR) as the criterion for tuning and for method evaluation.

We investigated light field inpainting and light field reconstruction from focal stack images. In all cases, we initialized the transform $\boldsymbol{T}$ with the $p_x \times p_y \times p_u \times p_v \times p_c$-point DCT and ran UTL for 120 iterations.

3.3.1 Inpainting Light Fields

We apply blind UTL to light field inpainting by minimizing (3.4) with $\boldsymbol{A} = \mathsf{diag}\{\mathsf{vec}(M)\}$ where

$$M[i,j,k,l,c] = \begin{cases} 1 & (i,j,k,l,c) \in \Omega \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

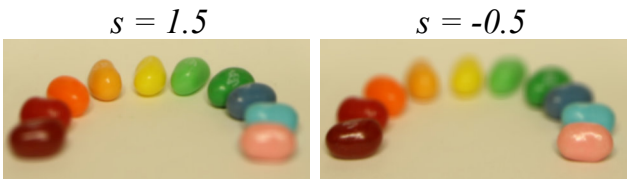


FIG 3.2 – An example 2-image focal stack and the amount of pixel shift, s, applied to light field before summing. Different jelly beans come into focus as the focal plane passes through the scene

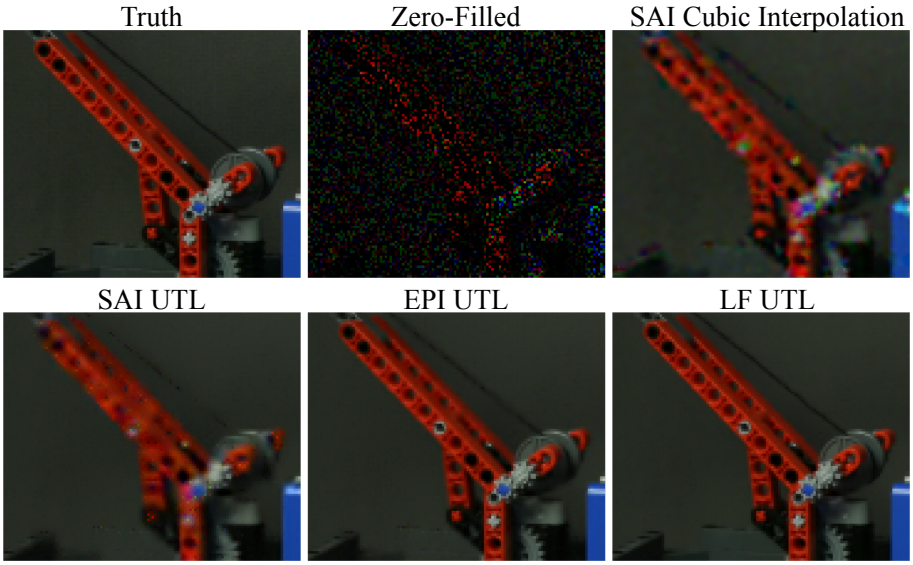


FIG 3.3 – Zoomed in view of the central perspective of inpainted Lego truck light fields.

and Ω is the set of samples taken of the light field. In our experiments, Ω is such that only 20% of all samples are kept at random. Samples in Ω were drawn independently for each light field tested. We used spatial cubic interpolation to initialize $\boldsymbol{x}$.

For the inpainting problem, we let $\lambda = 10^8$ and tuned the patch shape and sparsity threshold, γ, for all three patch shapes. In all cases, we used the full color patch dimension of 3. For SAI UTL and LF UTL, patch dimensions in x and y were constrained to be equal, while in LF UTL patch dimensions in u and v were similarly constrained. All methods had an upper bound on the largest patch that could be chosen due to memory constraints, as updating $\boldsymbol{T}$ in a blind setting precludes computing $\boldsymbol{Z}$ on-the-fly. While SAI UTL and EPI UTL did not reach that bound, and instead settled on smaller patch sizes, LF UTL did,

Method	n	Patch Shape $(p_x, p_y, p_u, p_v, p_c)$	γ
SAI UTL	108	(6, 6, 1, 1, 3)	0.0625
EPI UTL	135	(9, 1, 5, 1, 3)	0.0582
LF UTL	243	(3, 3, 3, 3, 3)	0.0454

TBL 3.1 – Patch Shape and Thresholds for inpainting problem. For brevity, we only list (x, u) patch dimension for EPI UTL, although we learn filters for the corresponding patches in (y, v) as well.

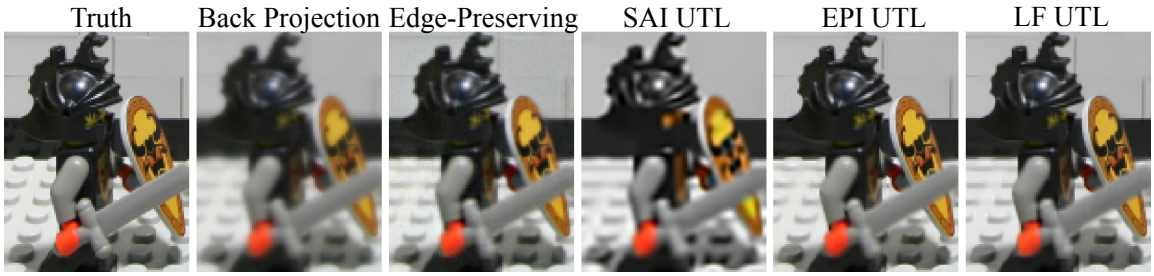


FIG 3.4 – Zoomed in view of the central perspective of Lego knight light fields reconstructed using different methods.

due to its increased dimensionality. **TBL** 3.1 lists the tuned hyperparameters for the three cases.

We applied each of the three methods with their tuned hyperparameters to the 10 light fields in our dataset. **TBL** 3.2 shows the PSNR of each of the reconstructions. LF UTL surpassed EPI UTL by 1.5dB and SAI UTL by 6.2dB on average. **FIG** 3.3 compares the performance of each of the methods on a zoomed in section of the Lego truck light field. LF UTL is able to preserve fine features more accurately than any of the other methods.

3.3.2 Reconstruction From Focal Stack Images

The capture of a photograph in a particular focal setting can be modeled by:

$$I_k(x, y, c) = \iint_{\mathcal{A}} L(x + s_k u, y + s_k v, u, v, c)\ \mathrm{d}u\ \mathrm{d}v, \tag{3.6}$$

	Cubic Interpolation		Proposed blind UTL methods		
light field	SAI	EPI	SAI UTL	EPI UTL	LF UTL
amethyst	28.50	26.74	31.72	37.67	**39.79**
crystal ball	21.56	21.29	25.68	30.22	**30.71**
Lego bulldozer	26.69	23.95	29.94	34.02	**35.63**
bunny	32.21	28.71	35.21	39.49	**42.04**
bracelet	23.28	22.84	27.99	34.70	**34.90**
eucalyptus	29.32	28.15	32.24	37.85	**39.69**
Lego knights	26.40	24.16	31.29	34.00	**36.68**
treasure chest	24.57	23.19	27.86	**32.92**	32.46
jelly beans	35.82	32.35	38.16	40.41	**41.81**
Lego truck	28.95	27.86	32.17	38.25	**40.62**
Average	27.73	25.92	31.23	35.95	**37.43**

TBL 3.2 – PSNR in dB for each recovered light field using different inpainting methods.

where $\mathcal{A}$ is the support set of the aperture, and s_k is a parameter determined by the focus setting of the kth photo. Thus, we can render photographs with a varying focal plane by appropriately adjusting s_k; see for example **FIG** 3.2. For further information regarding photograph capture and its relation to Fourier subspaces, see [59, 50].

Interestingly, (3.6) can also be written as

$$I_k(p,q,c) = \iiiint L(x,y,u,v,c)\,\delta_2((x-s_k u)-p,(y-s_k v)-q)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}u\,\mathrm{d}v\,. \quad (3.7)$$

Written this way, we can see that this forward model is a set of parallel plane integrals, similar to the Radon transform in CT.

We apply $\boldsymbol{A}_k$ by shifting subaperture images by s_k times their u, v coordinates and summing. $\boldsymbol{A}$ then represents the application of each $\boldsymbol{A}_k$ in a stack. We used linear interpolation to shift the subaperture images. Our measurement model is then

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\epsilon}, \quad (3.8)$$

where $\boldsymbol{\epsilon}$ denotes additive white Gaussian noise with standard deviation σ. In our experiments, we retrospectively added $\boldsymbol{\epsilon}$ with $\sigma = 1\%$ of the peak value of the photographs

	Scaled Back	Edge	Proposed blind UTL methods		
light field	Projection	Preserving	SAI UTL	EPI UTL	LF UTL
amethyst	28.94	33.39	29.11	36.19	**37.23**
crystal ball	20.33	23.30	21.83	24.65	**24.73**
Lego bulldozer	23.07	28.70	25.36	**30.01**	29.35
bunny	29.94	35.40	31.57	38.40	**39.59**
bracelet	18.23	24.46	23.18	**26.26**	24.51
eucalyptus	30.40	33.83	30.37	36.72	**37.56**
Lego knights	21.93	25.73	24.72	**28.10**	27.75
treasure chest	24.85	29.63	25.77	32.13	**32.15**
jelly beans	25.08	35.96	32.90	37.92	**38.44**
Lego truck	24.75	34.24	29.68	37.44	**38.42**
Average	24.75	30.46	27.45	32.78	**32.97**

TBL 3.3 – PSNR in dB for each light field using different focal stack reconstruction methods

$\boldsymbol{y}$. We used shift parameters $s \in \{-1, -0.5, 0, 0.75, 1.5\}$ to simulate 5 photographs taken with our model.

We compare our proposed method against an edge-preserving regularizer of the form:

$$\begin{aligned}\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} &\frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 \\ &+ \beta_{x,y} \sum_i \psi([\boldsymbol{C}_{x,y}\boldsymbol{x}]_i; \delta_{x,y}) \\ &+ \beta_{u,v} \sum_j \psi([\boldsymbol{C}_{u,v}\boldsymbol{x}]_j; \delta_{u,v}),\end{aligned} \tag{3.9}$$

where $\psi(\cdot, \delta)$ denotes the Huber potential function, a smooth approximation of an absolute value function. $\delta_{k,l}$ is a hyperparameter controlling the function curvature. $\boldsymbol{C}_{k,l}$ applies finite differences along dimensions k and l. We tuned $\beta_{x,y}$, $\delta_{x,y}$, $\beta_{u,v}$, $\delta_{u,v}$ on the same set as the UTL methods, which resulted in $7.47, 98.9, 3 \times 10^{-2}, 6 \times 10^{-4}$ for each parameter respectively.

For each of the UTL methods, we used the patch shape and threshold learned during inpainting and tuned λ, which resulted in $4.14 \times 10^{-2}, 2.61 \times 10^{-1}, 2.27 \times 10^{-1}$ for SAI, EPI, and LF UTL respectively. For the data update, we used 5 conjugate gradient iterations. **TBL** 3.3 shows the PSNR of each of the methods applied to the 10 light fields in our dataset.

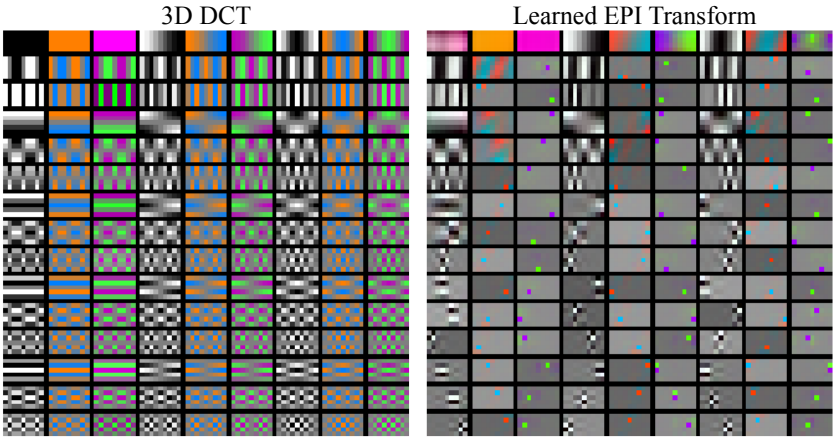


FIG 3.5 – Comparison of the filters learned using EPI UTL (right) with those of the 3D DCT (left) used to initialize the blind inpainting method. The learned filters adapt to the vertical linear structure of the EPI light field slices.

For this problem, LF UTL only out performed EPI UTL by 0.19dB on average. **FIG** 3.4 shows zoomed in views from the Lego knights light field.

3.4 Discussion

FIG 3.5 shows the reshaped rows of the transform learned on the epipolar patches of amethyst during blind inpainting. Each of these transform patches is effectively convolved with the light field for regularization. The filters have learned the mostly vertical linear structure of the epipolar domain by learning vertical finite-difference-like operations. We also see the slight tilt in some of the structures, reflecting the skew in out-of-focus pixels. As expected, most of this vertical structure is captured in luminance rather than color channels.

FIG 3.6 shows the filters learned using SAI UTL during blind inpainting. Similar to the EPI case, we find finite-difference-like structures in the luminance channels, but with less vertically aligned structure. In both cases, UTL learned shifted versions of the same filter. This is a weakness of the unitary constraint, because shifted versions of the same filter can be orthogonal, but provide no new information for regularizing the reconstruction. The unitary constraint also forces one to learn a low-pass filter, which one does not expect to induce sparsity in general. We leave it as future work to investigate more effective

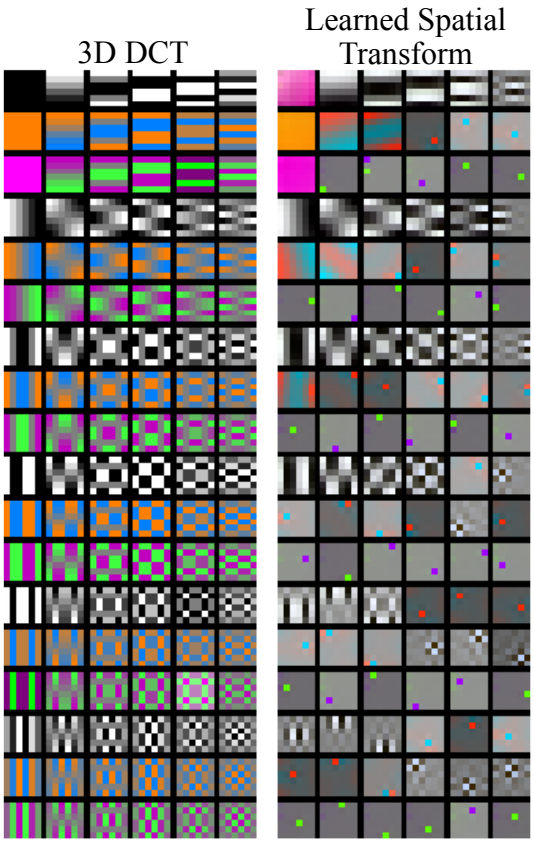


FIG 3.6 – Comparison of the filters learned using SAI UTL (right) with those of the 3D DCT (left) used to initialize the method.

constraints on the learned filters, such as Fourier magnitude incoherence [62] or tight-frame conditions on a wide transform, or some other consideration of light field physics.

We found that full dimensional patches best represented our data, but their increased dimensionality limited their extent in any one dimension due to memory constraints in storing $\boldsymbol{Z}$. As we assume many (but not all) of the rows of $\boldsymbol{Z}$ to be sparse, we believe UTL can be optimized for more efficient storage. An alternative may be to store a subsampled or sketch of $\boldsymbol{Z}$, and only approximate the $\boldsymbol{T}$ update. This work focused on maximally overlapping patches with a stride of 1, but larger strides could be used. We leave it as future work to investigate how these memory saving techniques impact reconstruction accuracy.

Because EPI patches were able to regularize the data nearly as well as full LF patches, presumably because of the shifting structure of the EPI dimensions, it would be interesting to see if a union of SAI and EPI transforms could capture the light field structure as well as full LF patches. Such unions have been effective in other inverse problems [97].

Combining adaptive sparsity with other regularizers such as low-rank models may also be effective [86].

3.5 Conclusion

This work investigated the effectiveness of using learned sparsifying transforms for different patch structures to regularize light field inverse problems. We found that full-dimensional patches provided the best data model, but EPI patches could capture most of the signal model with a lower dimensionality. We validated our proposed light field models on two inverse problems: light field inpainting and focal stack reconstruction. In both cases, regularization using transform learning yielded better reconstruction PSNR than simple handcrafted methods.

CHAPTER 4

Weighted Transform Learning for Light-Field Imaging

4.1 Introduction

In the previous chapter, we demonstrated that blind unitary transform learning (UTL) showed superior performance on compressive light-field imaging problems in comparison to other non-data-driven methods. The unitary constraint allowed us to avoid a trivial solution while also being computational efficient to compute, but unfortunately, was not motivated by what we would expect transform filters to look like. In fact, the unitary constraint requires that we learn a low-pass filter, despite such filters generally not producing sparse outputs on natural light fields and images. More generally, we may expect different filters to produce different levels of sparsity in UTL. In this work, we explore weighting the sparsity regularization of different filters. We investigate several methods for determining the weights, but find empirically that none of the proposed methods produce significant improvement over blind UTL.

Our work is most similar to that of Miyagi *et al.* [54], who weighted the filters of the DCT in a dictionary formulation with ℓ_1 sparsity for light field reconstruction. Because the DCT is a unitary operator, that approach is equivalent to the methods described here applied to a static DCT transform. Because we weight the sparse images instead of the filters, our weights are equivalent to the multiplicative inverse of the weights described in that paper. We also do not learn the weights on a dataset, and instead opt for a blind approach. This chapter further explores different weighting schemes and applies them to an adaptively estimated transform as opposed to a static one. As ℓ_0 sparsity is computationally feasible in a transform framework, we also opt for a ℓ_0 "norm" in our formulation to encourage sparsity.

The work in this chapter showed no significant improvements so was never published.

4.2 Methods

Recall from the previous chapter the blind UTL cost function for recovering an image $\boldsymbol{x}$ from measurement vector $\boldsymbol{y}$:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \min_{\boldsymbol{T}\in\mathcal{V}} \min_{\{\boldsymbol{z}_j\}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \left(\sum_j \|\boldsymbol{T}\boldsymbol{P}_j\boldsymbol{x} - \boldsymbol{z}_j\|_2^2 + \gamma^2 \|\boldsymbol{z}_j\|_0 \right). \tag{4.1}$$

The zero "norm" can be written in terms of a sum of indicator functions $\mathbb{I}_{c(q)}(q)$ which take the value 1 when the condition $c(q)$ is true and 0 otherwise

$$\|\boldsymbol{q}\|_0 = \sum_i \mathbb{I}_{q_i \neq 0}(q_i)\,. \tag{4.2}$$

We can generalize this "norm" by taking a weighted sum such that some indices are penalized more for being nonzero

$$\|\boldsymbol{q}\|_{\boldsymbol{w},0} = \sum_i w_i^2\, \mathbb{I}_{q_i \neq 0}(q_i)\,. \tag{4.3}$$

When using such a weighted sum, the corresponding proximal operator update applies a scaled hard thresholding for each element of $\boldsymbol{z}_j$

$$\mathsf{hard}(q_i; w_i\gamma) = \begin{cases} 0 & |q_i| \leq w_i\gamma \\ q_i & |q_i| > w_i\gamma. \end{cases} \tag{4.4}$$

From a filter bank perspective, such weighting allows each "arm" or channel of the filter bank to have its own threshold. Ideally, we would choose thresholds such that channels that carry more artifacts and noise have larger thresholds. It is not clear what the best choice of weights/thresholds would be. In this work we investigate several options for setting the weights in the context of blind transform learning. Because in the blind setting we do not assume to have training data, we attempt to infer good weights from a DCT transform $\boldsymbol{T}_{\text{DCT}}$, an initial reconstruction $\boldsymbol{X}_0$, or a better reconstruction like a edge-preserving regularized reconstruction $\boldsymbol{X}_{\text{EP}}$.

Weighting Option 1 One option for choosing the weights $\boldsymbol{w}$ was (equivalently) presented by Miyagi *et al.* [54] in the context of a weighted ℓ_1 norm for DCT regularization

$$w_i = \sqrt{\frac{1}{\frac{1}{J}\sum_j^J |\boldsymbol{T}_{\text{DCT}}\boldsymbol{X}_{\text{ref}}|_{i,j}^2}}. \tag{4.5}$$

In their work, they used training data $\boldsymbol{X}_{\text{ref}} \in \mathbb{R}^{n\times J}$ for learning the weights. In this work we will use patches from a EP reconstruction $\boldsymbol{X}_{\text{EP}}$. Here, if a filter does not sparsify the reference patches, then it gets assigned a small threshold. Conversely, if a filter sparsifies the image extremely well, then (4.5) assigns it a large threshold to encourage that sparsity at test time.

Weighting Option 2 A potential downside of the previous option is that it does not consider artifacts and noise when setting the thresholds. Intuitively, a channel that has more noise or artifact energy will need a larger threshold to remove these degradations. One way we can estimate where artifacts end up is by filtering the difference between a basic and more advanced reconstruction and increasing the weights where that energy ends up. One option for doing this could be

$$w_i = \sqrt{\frac{\frac{1}{J}\sum_j^J |\boldsymbol{T}_{\text{DCT}}(\boldsymbol{X}_{\text{EP}} - \boldsymbol{X}_0)|_{i,j}^2}{\frac{1}{J}\sum_j^J |\boldsymbol{T}_{\text{DCT}}\boldsymbol{X}_{\text{EP}}|_{i,j}^2}}. \tag{4.6}$$

Weighting Option 3 Consider the regularization term in the case of ℓ_1 sparsity where $\boldsymbol{W} = \mathsf{diag}\{\boldsymbol{w}\}$

$$\|\boldsymbol{T}\boldsymbol{X} - \boldsymbol{Z}\|_{\text{F}}^2 + \gamma\|\mathsf{vec}(\boldsymbol{W}\boldsymbol{Z})\|_1. \tag{4.7}$$

Performing a change of variable $\boldsymbol{Z} = \boldsymbol{W}^{-1}\tilde{\boldsymbol{Z}}$ yields

$$\left\|\boldsymbol{T}\boldsymbol{X} - \boldsymbol{W}^{-1}\tilde{\boldsymbol{Z}}\right\|_{\text{F}}^2 + \gamma\left\|\mathsf{vec}\left(\tilde{\boldsymbol{Z}}\right)\right\|_1. \tag{4.8}$$

If we assume we know $\boldsymbol{T}$,$\boldsymbol{X}$ and $\boldsymbol{Z}$, we can minimize this term with respect to each diagonal element of $\boldsymbol{W}^{-1}$ independently as

$$\hat{w}_{ii}^{-1} = \underset{w_{ii}^{-1}}{\arg\min} \left\|[\boldsymbol{T}\boldsymbol{X}]_{i,:} - w_{ii}^{-1}\tilde{\boldsymbol{Z}}_{i,:}\right\|_2 \tag{4.9}$$

$$= \frac{\langle \tilde{\boldsymbol{Z}}_{i,:}, [\boldsymbol{T}\boldsymbol{X}]_{i,:}\rangle}{\left\|\tilde{\boldsymbol{Z}}_{i,:}\right\|_2^2}. \tag{4.10}$$

FIG 4.1 – The Heidelberg Light Field Dataset Training Images, from left to right: Boxes, Cotton, Dino, and Sideboard. Only central views shown.

If we think of $\tilde{\boldsymbol{Z}}$ as sparse code corresponding to a true or clean image ($\approx \boldsymbol{T}_{\text{DCT}}\boldsymbol{X}_{\text{EP}}$), and $\boldsymbol{TX}$ as the noisy or corrupted sparse codes ($\approx \boldsymbol{T}_{\text{DCT}}\boldsymbol{X}_0$), then we might propose the following weights for our problem

$$w_i = \frac{\sum_j^J |\boldsymbol{T}_{\text{DCT}}\boldsymbol{X}_{\text{EP}}|_{i,j}^2}{\langle [\boldsymbol{T}_{\text{DCT}}\boldsymbol{X}_{\text{EP}}]_{i,:}, [\boldsymbol{T}_{\text{DCT}}\boldsymbol{X}_0]_{i,:} \rangle + \epsilon}. \tag{4.11}$$

Weighting Option 4 As one of the initial motivations for this study of weighted regularizers was to avoid sparsifying the low-pass filters that inherently must be learned by any unitary transform, we also could suggest this simple choice of weights that does not penalize the constant terms of the DCT (of which there are three in the case of color images)

$$w_i = \begin{cases} 0, & i \leq 3 \\ 1, & \text{otherwise.} \end{cases} \tag{4.12}$$

4.3 Experiments

We evaluated the proposed regularizer on light fields from the Heidelberg 4D Light Field Dataset [38]. We used the 4 "Training" light fields from the dataset (namely, Boxes, Cotton, Dino, and Sideboard, see FIG 4.1) to set hyperparameters β and γ and to choose from the proposed weighting choices. Once these parameters were chosen, the method was tested on 19 light fields from the "Test" and "Additional" collections of the dataset. For all light fields, we extract the central 5×5 views.

For all experiments, we jointly reconstructed and demosaiced light fields from Bayer-encoded focal stack measurements. As the light fields in this dataset have SRGB color, we first transform the light fields into a linear RGB color space before any processing.

Method	β	γ	Avg. NRMSE
UTL	7.954	0.017770	0.0531
(4.5)	7.921	0.003246	0.0580
(4.6)	7.590	0.036573	0.0520
(4.11)	12.993	0.008254	**0.0507**
(4.12)	13.749	0.012313	0.0518

TBL 4.1 – Tuned regularization parameters for weighted UTL. Weighted UTL with weights chosen by (4.11) performed the best on average for the 4 training light fields.

We computed quality metrics, such as PSNR, in the (perceptual) SRGB space. We generate measurements at the original 512×512 resolution, downsample by 2, and add noise so our measurements have 50dB PSNR. We reconstruct at 256×256 spatial resolution and downsample the dataset light fields by 2 for comparison. By generating our measurement with higher resolution light fields, we avoid committing an "inverse crime" in our experiments [18].

We initialized all methods by bilinear demosaicing the focal stack photographs and then back projecting the result through the focal stack model. For patches, we extract 3×3 spatial patches with a stride of 1 and maintain full dimensionality in other dimensions resulting in a patch shape of $5 \times 5 \times 3 \times 3 \times 3$. Our measurement model generated 256×256 Bayer patterned sensor images photographs via the shift-and-sum method (see Section 3.3.2) with pixel shifts $-1, -0.5, 0, 0.75, 1.25$ and bilinear interpolation.

For implementing the edge-preserving (EP) reconstruction, we used a Huber potential with the parameters described on page 30 and ran 50 iterations of conjugate gradient which seemed adequate. We ran UTL and the proposed methods for 100 iterations using the BCD approach described previously in Section 2.5.2.1.

We tuned β, γ, and the weighting choice over 1000 trials with the objective of minimizing the average NRMSE of the 4 training images. For fair comparison, UTL was also similarly tuned. **TBL** 4.1 summarizes the results of the parameter search. Weighted UTL with weights chosen by (4.11) performed the best on average for the 4 training light fields.

With the selected parameters, we ran EP, UTL and the proposed weighted UTL on the 19 light fields in the test set. **TBL** 4.2 shows the PSNR for each reconstructed light field. The proposed algorithm improved the average PSNR of the test set by just under 0.2dB.

Light Field	EP	UTL	WUTL
antinous	35.116	35.687	**37.189**
bedroom	27.080	**30.664**	30.644
bicycle	25.107	**28.620**	28.568
boardgames	25.021	28.819	**29.092**
dishes	26.234	**28.664**	28.554
greek	31.248	33.429	**33.824**
herbs	29.179	30.694	**30.913**
kitchen	26.364	30.942	**31.042**
museum	29.126	**32.888**	32.883
origami	25.719	**29.479**	29.350
pens	30.771	32.239	**32.403**
pillows	30.578	32.466	**32.757**
platonic	28.993	**31.245**	31.160
rosemary	28.674	31.485	**31.533**
table	26.725	**31.050**	30.962
tomb	35.369	35.100	**36.141**
tower	30.545	**31.311**	31.159
town	28.129	**32.317**	32.146
vinyl	27.689	**32.301**	32.251
Average	28.825	31.547	**31.714**

TBL 4.2 – PSNR in dB for weighted UTL on Heidelberg light field dataset

Just over half of the light fields were better recovered using plain UTL, though some light fields showed more than 1dB of improvement with weighted UTL.

4.4 Conclusion

While the proposed weighted UTL approach showed significant improvement on a few test light fields, our results show that the improvement is not significant on average when regularization parameters are tuned carefully. It is unclear if there is any systematic way to predict which of either UTL or weighted UTL will perform better for any particular dataset.

Since plain UTL corresponds to a special case of weighted UTL, for any one test case there exists an optimal weighting that does no worse then plain UTL. This work attempted to approximate this optimal weighting in a non-rigorous fashion with some simple weighting formulations that adapted to each test case. While the optimal weighting could do no worse than plain UTL, it is unclear if it would do significantly better. Thus, our poor results could be due either to poorly finding good weightings for each test image or because the method is inherently of little help to the reconstruction. One observation which might suggest the latter is that the low-pass filter's sparse codes are generally much larger than the thresholds used in plain UTL. Thus, scaling down those thresholds (or even setting them to zero) does not affect the results much in the case of hard thresholding. We might expect more of a difference with ℓ_1 sparsity regularization, but that regularizer generally performs worse than ℓ_0 sparsity regularization.

The average PSNR improvement shown here is not large enough for us to confidently recommended weighted UTL. It is likely that applying the weighting schemes considered here could degrade PSNR for any given test case.

The weighting methods investigated herein are all "hand crafted" formula and it might be possible to develop a machine-learning approach for learning a mapping from the raw data to the weights given sufficient training data. For example, Gu *et al.* [33] found a supervised choice of tuning parameters could help ℓ_1-sparse MRI reconstruction perform as well as deep learning. Perhaps a neural network could better estimate these weights. However, if such training data is available then perhaps the transform $\boldsymbol{T}$ could also be learned in a supervised manner, rather than in the "blind" learning style considered here. On the other hand, some combinations of blind learning and supervised learning have been shown to be beneficial [45].

CHAPTER 5

Generalizability of Learned Unitary Transforms

5.1 Introduction

Unitary Transform Learning comes in two distinct variants. In standard non-blind (population-adaptive) UTL, the transform — aka filter — coefficients are trained to fit a population of training data under the assumption that members of the population will be sparsifiable under a common transform. This training data is assumed to be representative of the unknown data that is going to be reconstructed. If the image to be reconstructed has features similar to the training data, then we can expect the transform to sparsify it well, leading to greater noise rejection and higher reconstruction accuracy. For these same reasons, if the training data is not representative, we would expect reconstruction accuracy to decrease. It is therefore important to have representative training data whenever applying a population-adaptive approach like non-blind UTL.

In contrast with non-blind UTL, blind (instance-adaptive) UTL requires no training data. Instead, blind UTL learns a transform at test time in an instance-adaptive fashion. As the reconstruction iterates, the transform should be refined as better and better estimates of the final image are produced. Note the transform no longer represents sparsifiable features of a population, but now merely enforces the sparsifiability of the reconstructed image. In denoising problems, where noise is often modeled as independent in image space, it is reasonable to expect the noise to not be sparsifiable. In practice, this model works well even for more complicated inverse problems.

Thus, despite their clear structural similarity, non-blind and blind UTL represent different assumptions about our signal, sparsifiable under a population specific transform and just sparsifiability, respectively. As a population-specific transform is more constrained

The work in this chapter showed no significant improvements so was never published.

and brings in information about the population in the form of sparsifying filters, we might hypothesize that non-blind UTL would out perform blind UTL in general. But this dependence on a population might make non-blind UTL more prone to error when reconstructing out-of-population samples, such as anomalies or never before seen anatomies or objects.

It is therefore unclear when to prefer non-blind or blind UTL in general. This chapter aims to answer the following question: How representative does your training data need to be to prefer non-blind UTL over the more adaptive blind UTL ? *I.e.*, how well can a unitary transform generalize to dissimilar data? As a concrete problem, this chapter examines the reconstruction accuracy of T1-weighted brain and proton-density knee MRI images with transforms trained on a varying mix of both populations.

5.2 Related Works

Unitary transform learning in the context of MRI reconstruction was first proposed by Ravishankar and Bresler [67], but they did not compare non-blind and blind version or suggest a strategy to pick one over the other. Bahadir *et al.* [6] used data-driven methods to design sampling patterns for MRI, and they compared the optimal sampling patterns for knee and brain images. Unlike this work, they did not look at the effect anatomy has on the reconstruction algorithm.

5.3 Unitary Transform Learning

Section 2.5 provides thorough description of transform learning. For completeness, we provide a brief review of population-adaptive (non-blind) and instance-adaptive (blind) unitary transform learning.

In non-blind unitary transform learning, we first learn a unitary transform in a training phase. The objective of this training is to minimize the expected fit of image patches $\boldsymbol{P}_j\boldsymbol{x}$ over a population of images $\{\boldsymbol{x}_1 \dots \boldsymbol{x}_K\}$ with respect to the transform $\boldsymbol{T}$ and sparse codes $\boldsymbol{z}$:

$$\hat{\boldsymbol{T}} = \underset{\boldsymbol{T}\in\mathcal{V}^{n\times n}}{\arg\min} \min_{\{\boldsymbol{z}_{j,k}\}} \frac{1}{K}\sum_{k=1}^{K}\sum_{j} \|\boldsymbol{T}\boldsymbol{P}_j\boldsymbol{x}_k - \boldsymbol{z}_{j,k}\|_2^2 + \gamma\|\boldsymbol{z}_{j,k}\|_0. \tag{5.1}$$

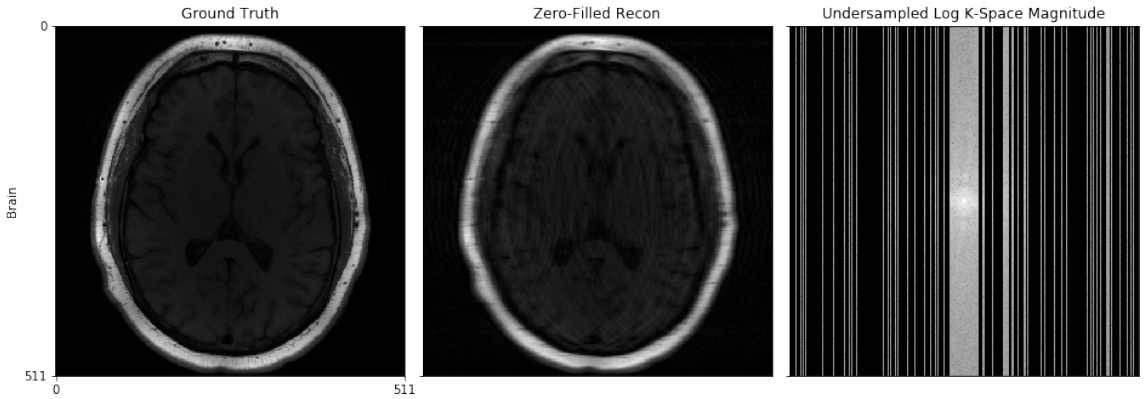


FIG 5.1 – Example brain image reconstructed with ESPIRiT [46] and SENSE on fully sampled k-space, zero-filled undersampled reconstruction, and the log undersampled k-space

For notational convenience, we let $\boldsymbol{X} = [\boldsymbol{P}_1\boldsymbol{x}_1 \dots \boldsymbol{P}_J\boldsymbol{x}_1 \dots \boldsymbol{P}_J\boldsymbol{x}_K]$ and similarly $\boldsymbol{Z} = [\boldsymbol{z}_{1,1} \dots \boldsymbol{z}_{J,1} \dots \boldsymbol{z}_{J,K}]$. We can then simplify (5.1) as

$$\hat{\boldsymbol{T}} = \underset{\boldsymbol{T}\in\mathcal{V}^{n\times n}}{\arg\min} \min_{\boldsymbol{Z}} \frac{1}{K}\|\boldsymbol{T}\boldsymbol{X} - \boldsymbol{Z}\|_F^2 + \gamma^2\|\mathsf{vec}(\boldsymbol{Z})\|_0. \tag{5.2}$$

We minimize (5.2) via block-coordinate descent on $\boldsymbol{T}$ with a Procrustes update and on $\boldsymbol{Z}$ with hard-thresholding [67].

Once a suitable estimate of $\boldsymbol{T}$ has been obtained, it can be used in iterative optimization routines to reconstruct images at test time via this regularizer on each image patch:

$$\mathsf{R}(\boldsymbol{x}) = \min_{\{\boldsymbol{z}_j\}} \sum_j \left\|\hat{\boldsymbol{T}}\boldsymbol{P}_j\boldsymbol{x} - \boldsymbol{z}_j\right\|_2^2 + \gamma^2\|\boldsymbol{z}_j\|_0. \tag{5.3}$$

Blind UTL, on the other hand, has no training phase, and we just apply the following regularizer in the reconstruction of each image $\boldsymbol{x}$:

$$\mathsf{R}(\boldsymbol{x}) = \min_{\boldsymbol{T}\in\mathcal{V}^{n\times n}} \min_{\{\boldsymbol{z}_j\}} \sum_j \|\boldsymbol{T}\boldsymbol{P}_j\boldsymbol{x} - \boldsymbol{z}_j\|_2^2 + \gamma^2\|\boldsymbol{z}_j\|_0. \tag{5.4}$$

Updates of blind-UTL follow that of the non-blind version: a block coordinate descent on $\boldsymbol{x}$, $\boldsymbol{T}$, and each $\boldsymbol{z}_j$ (see Chapter 2).

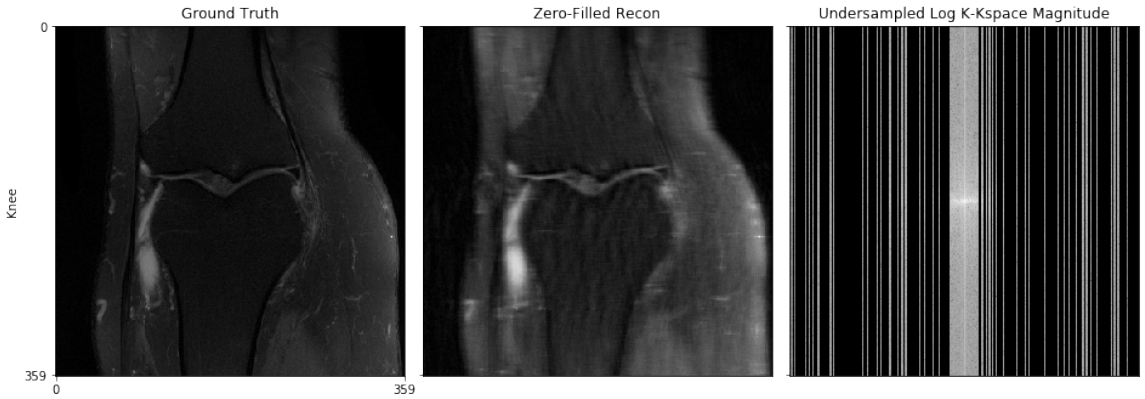


FIG 5.2 – Example knee image reconstructed with ESPIRiT [46] and SENSE on fully sampled k-space, zero-filled undersampled reconstruction, and the log undersampled k-space

5.4 Experiments

To better understand the affect of training data mix, we compared the reconstruction accuracy of non-blind UTL on test sets consisting of brain MRI images and knee MRI images after being trained on a population of varying amounts of each anatomy. We used (5.1) to train UTL and applied our learned transforms for each trial t by minimizing the following regularized cost function

$$\hat{\boldsymbol{x}}_t = \arg\min_{\boldsymbol{x}} \min_{\{\boldsymbol{z}_j\}} \lambda \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \sum_j \left\| \hat{\boldsymbol{T}}_t \boldsymbol{P}_j \boldsymbol{x} - \boldsymbol{z}_j \right\|_2^2 + \gamma_{\text{test}}^2 \|z_j\|_2. \tag{5.5}$$

The system matrix $\boldsymbol{A}$ represents the phase-encoded undersampled Fourier transform and $\boldsymbol{y}$ represent the undersampled k-space data. Hyperparameter λ encodes a trade-off in data-fit and model-fit. DCT reconstructions were performed by letting $\boldsymbol{T}_t$ in (5.5) be the 2D-DCT transform. Blind UTL reconstructions were performed by additionally minimizing over $\boldsymbol{T}_t$ each iteration.

The dataset consists of 512×512 T1-weighted FLAIR axial brain images (**FIG** 5.1) and 360×360 proton-density coronal knee images (**FIG** 5.2). Gold standard (complex-valued) data was reconstructed using ESPIRiT sensitivity maps in a SENSE reconstruction [46]. The original knee data was collected with 15 channels on a 3T Siemens scanner [92], while the original brain data was collected with 32 channels on a 3T GE scanner. The gold standard data was then retrospectively undersampled by masking fully-sampled k-space with a randomly-generated mask to simulate $4\times$ single coil phase encode acceleration. A total of 100 2D slices were used in the study, 50 from brains and 50 from knees. The brain images

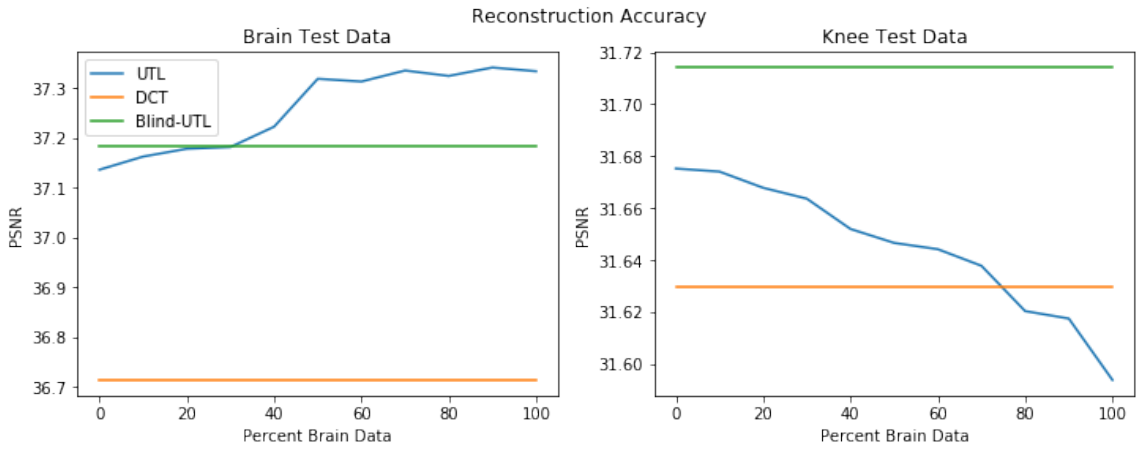


FIG 5.3 – Reconstruction accuracy as a function of training mix. (Left) UTL's accuracy on brain images improves as the amount brain training data increases. (Right) UTL's accuracy on knee images improves as the amount knee training data increases. However, in both cases the difference only spans 0.1dB.

and knee images were split evenly between train, validation and test sets for a total of 40, 20, and 40 slices respectively. In any one trial training non-blind UTL, 20 slices were used from the training dataset varying between 20 brain images to 20 knee images to simulate different mixes.

The validation dataset was used to tune γ_{train} and γ_{test}. While these perform a similar role in the training phase and the reconstruction/test phase, empirical results show that better accuracy can be achieved with different values. This may be because the test data is subject to greater noise than the training data. We used the black-box Bayesian tree-structured Parzen estimator (TPE) [8] method to tune these hyperparameters jointly on brain data and knee data over the course of several days. From this we found $\gamma_{\text{train}} = 0.156$, $\gamma_{\text{test,brain}} = 0.0212$ for brain data and $\gamma_{\text{test,knee}} = 0.0305$ for knee data. In all cases we fixed λ to be 10^8, as the data was noiseless, and we wanted to retain sampled values, effectively enforcing an $\boldsymbol{Ax} = \boldsymbol{y}$ constraint.

Unitary transforms were then learned using non-blind UTL on 11 different training dataset mixes ranging from 0% to 100% brain images (*i.e.*, at 70% brain images, the training set consisted of 14 brain images and 6 knee images). Once a transform was learned it was applied as a transform sparsity regularizer to reconstruct 20 brain images and 20 knee images in the test set. **FIG** 5.3 shows the test reconstruction accuracy as a function of training set mix for both brains and knees. **FIG** 5.3 also shows the results of adaptive

blind UTL and standard DCT transforms (both of which do not depend on training data) for comparison.

5.5 Discussion

The results in FIG 5.3 show that for brain images, we need about 30% brain images in our population to outperform the blind method, with little improvement after 50% brain data. Overall, using representative brain data boosted UTL by 0.1dB over blind UTL and by 0.6dB over the non-adaptive DCT.

For the knee test images, we needed about 30% knee training data to outperform the non-adaptive DCT. Surprisingly, for this population, the blind UTL method outperformed the learned method even when the training population was 100% knee data. Overall though, the spread across training mixes was less than 0.1dB.

Qualitatively speaking, compared to the knee data, the brain images seem more similar to each other. The round structure of the brain anatomy makes the edge content of the image more robust to slight rotations and is more similar across subjects. The knee images on the other hand vary in bone size, fat content and joint angle. For these variations, it seems that a square unitary transform does not have the degrees of freedom to adequately model the population.

Interestingly, blind UTL only outperformed the DCT by 0.5dB on the brain data and even less on the knee data. Both methods are effective at removing aliasing artifacts from the black background and so get a similar boost in PSNR in early iterations on MRI data. Even after masking the results to relevant structure, the performance is similar. In fact, many of the PSNR curves presented in the original blind UTL paper [67] on MRI data are nearly identical[1] when run with a fixed DCT transform — an observation that presumably the authors [67] did not realize at the time of its publication. It seems that the DCT is a very good transform for MRI data and larger improvements are only seen when the data is drawn from a sufficiently different distribution. For example, blind UTL vastly outperforms a fixed DCT on piecewise constant data as it is able to learn finite-difference-like filters. This is the benefit of adaptive regularization; we do not need to know whether the data will be better sparsified by something DCT-like or finite-difference-like as blind UTL will pick the superior filter.

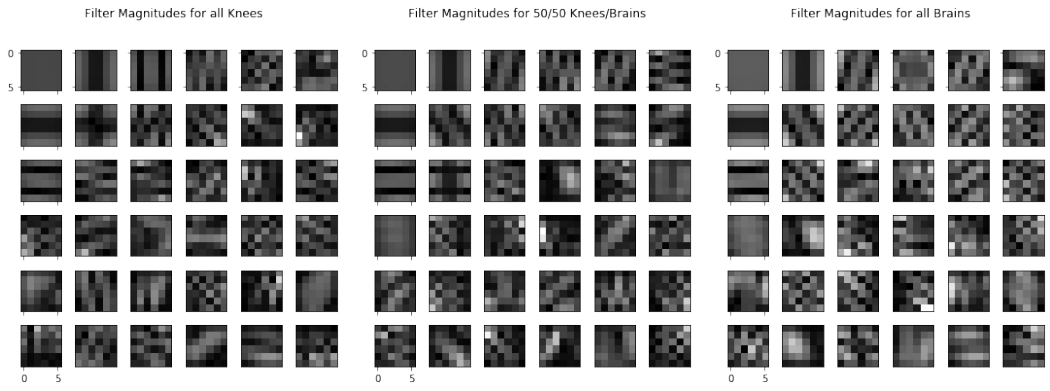


FIG 5.4 – Left: Magnitude of learned complex filters on all knees training data. Center: Magnitude of learned complex filters when half of the training data were knees and half were brains. Right: Magnitude of learned complex filters on all brains training data.

FIG 5.4 presents the magnitude of the complex filters as a function of training mix. The filters for brain images look tuned to find edges at a variety of angles, while the knee filters are harder to interpret.

5.6 Conclusion and Future Work

The work set out to understand when it is advantageous to train a unitary transform from *a priori* training data, over opting for a blind method that learns a unitary transform in an instance adaptive fashion. The unexpected conclusion of this preliminary study has been that pre-learning is only advantageous when the training data is highly correlated and self similar. Large variations in training populations are too diverse to be sufficiently captured in a unitary model. In a blind setting, a unitary transform only has to sparsify a single image estimate, and so the limited expressibility of a unitary transform is less of a hindrance.

Traditionally, the unitary constraint has been motivated by the need to avoid a trivial solution $\boldsymbol{T} = \boldsymbol{0}$ and $\boldsymbol{Z} = \boldsymbol{0}$. While in the absence of the unitary constraint, this is clearly a global minimizer, it is unclear whether BCD would find this global minimizer given an arbitrary initialization due to the non-convexity of the problem. **FIG** 5.5 shows a preliminary result of fitting a transform to an image without the unitary constraint, with a DCT initialization. The transform update becomes a simple least-squares solution. Preliminary efforts at applying this transform in a non-blind setting suggest that the local minimizer

[1]per iteration, comparing elapsed time, the DCT rises faster since no SVD is computed.

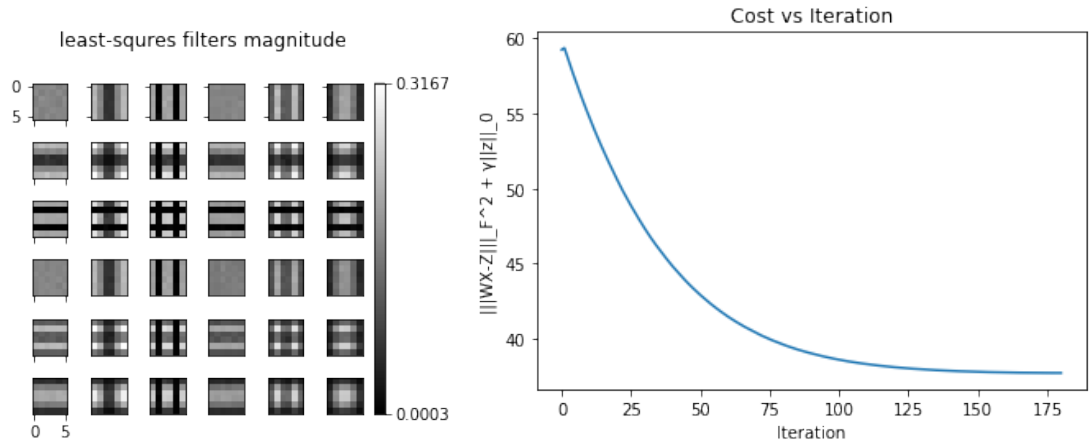


FIG 5.5 – Preliminary result of a transform filters learned using a least squares transform update on a single image. (Left) the filter magnitudes (Right) the cost $\|\boldsymbol{T}\boldsymbol{X} - \boldsymbol{Z}\|_{\mathrm{F}}^2 + \gamma \|\mathsf{vec}(\boldsymbol{Z})\|_0$ converges to a nonzero fixed point from a DCT initialization.

attained in this way with BCD is no more effective than the minimizer found with the unitary constraint.

These results highlight two possible paths for future work for improving the expressibility of transform models. (1) The minimizers found by BCD may be suboptimal, and it may be beneficial to explore joint updates of $\boldsymbol{T}$ and $\boldsymbol{Z}$. Currently, $\boldsymbol{T}$ is fit to $\boldsymbol{Z}$ and $\boldsymbol{Z}$ is fit to $\boldsymbol{T}$ repeatedly. This interaction may cause the BCD updates not to stray too far from the initialization and tend to converge slowly. (2) Minimizing the sparsity fit may not be the most effective training model. Equation (5.1) proposes we choose a unitary transform that minimizes the expected transform sparsity error. Preliminary results (**FIG** 5.5) without the unitary constraint suggest that this formulation may be suboptimal even without the unitary constraint. It may be more effective to minimize expected test time reconstruction error directly (a true supervised loss), using a combination of backpropagation and subgradient descent on a bilevel optimization problem [20]

$$\bar{\boldsymbol{W}} = \arg\min_{\boldsymbol{W}} \mathbb{E}_{\boldsymbol{x},\boldsymbol{\epsilon}} \left[\|\boldsymbol{x} - f(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{\epsilon}, \boldsymbol{W}, \gamma)\|_2^2 \right] \tag{5.6}$$

where f applies a set number of iterations of transform regularized image reconstruction. This formulation naturally drops the need for a unitary constraint, as the trivial solution, $\boldsymbol{T} = \boldsymbol{0}$, $\boldsymbol{Z} = \boldsymbol{0}$, would generally not be optimal for reconstruction accuracy.

In light of this, a promising future direction would be to replace the limited expressibility of a transform with a multilayer (deep) transform model updated by backpropagation and subgradient descent.

CHAPTER 6

Deep Generative Regularization on Patches

6.1 Introduction

The demand for quality high-resolution sampling in spatial, temporal, spectral and other physical parameter spaces in a variety of sensing applications, such as CT, MRI, remote sensing and light-field imaging, has led to a need to sample at sub-Nyquist frequencies to fit a fixed sampling budget. As a result, the traditional reconstructions are generally corrupted or not directly interpretable.

A common way to handle these inverse problems is to pose the reconstruction as an optimization problem consisting of *data-fidelity* terms that enforce our expectation that our collected samples are consistent with our reconstruction given a reasonable amount of measurement noise, and *regularization* terms, that enforce our measurement-independent expectation about the class of images we are reconstructing.

The undersampled nature of the measurements generally makes solving the data-fidelity term alone ill-posed. As such, developing effective regularization is essential to image-reconstruction accuracy. Developing effective regularization is challenging, as it must provide a model for the true signal that is both flexible enough to represent all plausible true images, yet discriminating enough to reject noise and artifacts.

A common approach in the design of regularization is modeling redundant features with reduced dimensionality. An image, for example, may have repeating or similar textures that, when extracted into image patches, can be more easily modeled, such as belonging to a union of subspaces. Many signals of interest fall into a shift-invariant class (at least locally). Shifting an image by a few pixels generally produces an image that is equally realistic. By modeling overlapping data patches instead of entire images, we implicitly model the shift-invariant structure of images.

The work in this chapter showed no significant improvements so was never published.

Classically, subspaces have been used to describe patch-wise models. These models are attractive as they dramatically simplify analysis. But this simplicity also results in a lot of reduced model flexibility. For this reason, nonlinear models have been growing in popularity, especially with modern deep neural networks.

Modern generative networks show remarkable performance on synthesizing fairly realistic images from low-dimensional latent spaces [63]. These networks tend to be highly parameterized and require large datasets to train on realistic image sizes.

6.2 Related Work

Recently, Bora *et al.* [12] applied generative models in the context of compressed sensing (CSGM) with the following cost function (*c.f.* [12, (1, 3)])

$$\hat{\boldsymbol{x}} = G(\hat{\boldsymbol{z}}), \quad \hat{\boldsymbol{z}} = \arg\min_{\boldsymbol{z}} \|\boldsymbol{A}G(\boldsymbol{z}) - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{z}\|_2^2, \tag{6.1}$$

where G is a nonlinear generative function generally resulting from training a neural network. As many generative networks impose a Gaussian assumption on the distribution of $\boldsymbol{z}$, (6.1) penalizes the ℓ_2 norm of $\boldsymbol{z}$, which is consistent with the Bayesian interpretation of the problem. This model leads to an estimate $\hat{\boldsymbol{x}}$ that lies in the range of the generator $\mathcal{R}(G)$, which may be unnecessarily restrictive and may limit any features not present in the training data from being seen. The nonlinearity of the generator makes optimizing $\boldsymbol{z}$ prone to converging to poor local minima when minimized via a standard subgradient descent. Bora *et al.* [12] circumvented this problem by rerunning their algorithm with 10 random initializations and choosing the solution with the least measurement error.

Building on the work of Bora *et al.*, Shah and Hegde [72] reformulated (6.1) as a constrained optimization problem of the form

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\in\mathcal{R}(G)} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 \tag{6.2}$$

and solved it using projected gradient descent (PGD-GAN). This approach reportedly improved performance and allowed for convergence guarantees, but the projection requires an inner minimization. To overcome this challenge, Raj *et al.* [64] replaced the projection minimization with a learned projection network based on a learned nonlinear "pseudoinverse" of G. Several other works have looked at learning image projectors outside the context of the range space of a generator network [15].

In comparison to those previously described works that all depend on a pretrained generator, the Deep Image Prior (DIP) method aims to fit an *untrained* generator to an image by fixing its input $\boldsymbol{z}$ and optimizing its weights $\boldsymbol{\theta}$ at reconstruction time [82]

$$\hat{\boldsymbol{x}} = G(\boldsymbol{z}; \hat{\boldsymbol{\theta}}), \quad \hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \|\boldsymbol{A}G(\boldsymbol{z}; \boldsymbol{\theta}) - \boldsymbol{y}\|_2^2. \tag{6.3}$$

While a generative CNN could in theory produce corrupted or noisy images, Ulyanov *et al.* [82] note that it learns smooth representations more quickly. Thus, early stopping the optimization can achieve a cleaner image. Heckel and Hand [37] further improved on this work by proposing an architecture with a more limited capacity that was more robust to fitting noise and artifacts.

Those methods required the entire image to lie in the range space of a generator. While likely requiring more training data to produce realistic output in the trained methods, such approaches also fix the dimensions of the image $\boldsymbol{x}$. Thus, the model cannot easily be used in similar inverse problems without first reparameterizing and, if trained, retraining the network. Patch-based models require less data to train and can easily be applied to varying image sizes. Gilton *et al.* [30] investigated the use of patch-based denoising networks in both an unrolled optimization and RED framework and found that they improved performance in small-dataset settings.

This chapter investigates the effectiveness of using generative models for regularization applied patch-wise.

6.3 Method

We investigated both methods that rely on a pretrained generator, as well as methods that adaptively learn a generator's weights at test time. This distinction is similar to that of population-adaptive and instance-adaptive (blind) UTL, where the former learns a transform based on a dataset (population) and the latter learns a transform at test time. Here we divide the proposed methods into these two categories, pretrained generators and blind generators.

6.3.1 Pretrained Generator

Our model for a pretrained generator can be formulated as

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta\, \mathsf{R}(\boldsymbol{x}) \tag{6.4}$$

$$\mathsf{R}(\boldsymbol{x}) = \min_{\{\boldsymbol{z}_j\}} \sum_j \|\boldsymbol{P}_j\boldsymbol{x} - G(\boldsymbol{z}_j)\|_p^p \tag{6.5}$$

where $\boldsymbol{x}$ denotes the unknown image, $\boldsymbol{y}$ the given measurements, $\boldsymbol{A}$ the system model encoding the measurement physics, $\boldsymbol{P}_j$ extracts the jth patch from $\boldsymbol{x}$, and $p \in \{1, 2\}$. The choice $p = 2$ is popular for its ease of optimization, but $p = 1$ has shown robustness to outliers, better generalization performance in neural network losses [96], and gives the system model more relative weight on outliers.

BCD provides a convenient way to minimize (6.4). We update $\boldsymbol{z}$ by computing the subgradient of $\boldsymbol{z}$ w.r.t. the cost using backpropagation and minimizing till convergence with ADAM [44]. For $p = 2$, we get the following update for $\boldsymbol{x}$

$$\hat{\boldsymbol{x}} = \left(\boldsymbol{A}'\boldsymbol{A} + \beta \sum_j \boldsymbol{P}_j'\boldsymbol{P}_j\right)^{-1} \left(\boldsymbol{A}'\boldsymbol{y} + \beta \sum_j \boldsymbol{P}_j' G(\boldsymbol{z}_j)\right). \tag{6.6}$$

For large problems where constructing $\boldsymbol{A}'\boldsymbol{A}$ is prohibitive, conjugate gradient could be used for the $\boldsymbol{x}$ update.

In the case of $p = 1$, deriving an efficient optimization method for the $\boldsymbol{x}$ update is less straight forward. For the results shown below I used a generic (accelerated) subgradient method, specifically ADAM [44], as I did with the $\boldsymbol{z}$ update. Nonetheless, here I will consider more specific algorithms that one could further develop for the ℓ_1 case. If patches are taken without any overlap, then $\beta\,\mathsf{R}$ has (for fixed $\boldsymbol{z}$) a simple proximal operator

$$\mathrm{prox}_{\beta\,\mathsf{R}}(\boldsymbol{x}) = \mathsf{soft}(\boldsymbol{x} - \bar{\boldsymbol{x}}, \boldsymbol{w}\beta) + \bar{\boldsymbol{x}} \tag{6.7}$$

where $\bar{\boldsymbol{x}} = \sum_j \boldsymbol{P}_j' G(\boldsymbol{z}_j)$ and $\boldsymbol{w}$ is an indicator vector with 1s in the ith position if $\boldsymbol{x}_i$ is included in a patch (commonly all pixels are included in one of the patches and so $\boldsymbol{w}$ can be disregarded). This proximal operator can be used in efficient methods such as POGM [78, 43].

When patches are overlapping such that every pixel appears in n patches, we can segment the J patches into n sets $J_1 \dots J_n$. We can then write R (given $\boldsymbol{z}$) as

$$\mathsf{R}(\boldsymbol{x}; \boldsymbol{z}) = \sum_{i=1}^{n} \left\| \boldsymbol{x} - \sum_{j=1}^{J_i} \boldsymbol{P}_j' G(\boldsymbol{z}_j) \right\|_1 \tag{6.8}$$

$$= \sum_{i=1}^{n} \|\boldsymbol{x} - \bar{\boldsymbol{x}}_i\|_1. \tag{6.9}$$

Interestingly, the minimizer (w.r.t. $\boldsymbol{x}$) of R in this case is the voxel-wise medians of the n images. Finding its proximal operator is more complicated, and involves sorting the pixels at each position between the n images and computing the proximal operator based on the index of each pixel of $\boldsymbol{x}$ in the sorted pixels. This would be computationally intensive, but element-wise separable w.r.t. $\boldsymbol{x}$ and thus highly parallelizable.

It is unclear to me whether the proximal-operator–based approach described above would be favorable over a splitting-based approach, which could be formulated as

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \min_{\{\boldsymbol{w}_i, \boldsymbol{\eta}_i\}} \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \beta \sum_{i}^{n} \|\boldsymbol{w}_i\|_1 + \frac{\mu_i}{2} \left(\|\boldsymbol{x} - \bar{\boldsymbol{x}}_i - \boldsymbol{w}_i + \boldsymbol{\eta}_i\|_2 - \|\boldsymbol{\eta}_i\|_2^2 \right). \tag{6.10}$$

ADMM would then involve the following updates (not including $\boldsymbol{z}$ update)

$$\boldsymbol{x} = \left(\boldsymbol{A}'\boldsymbol{A} + \beta \sum_{i}^{n} \mu_i \right)^{-1} \left(\boldsymbol{A}'\boldsymbol{y} + \sum_{i}^{n} \bar{\boldsymbol{x}}_i + \boldsymbol{w}_i - \boldsymbol{\eta}_i \right) \tag{6.11}$$

$$\boldsymbol{w}_i = \mathsf{soft}\left(\boldsymbol{x} - \bar{\boldsymbol{x}}_i + \boldsymbol{\eta}_i, \frac{\beta}{\mu_i} \right) \quad \forall i \tag{6.12}$$

$$\boldsymbol{\eta}_i = \boldsymbol{\eta}_i + \boldsymbol{x} - \bar{\boldsymbol{x}}_i - \boldsymbol{w}_i \quad \forall i \tag{6.13}$$

which is at least n-way parallelizable in the proximal update. The results presented in this chapter will use a subgradient method, but in future work, one could further investigate these methods for the ℓ_1 norm.

6.3.2 Blind Generator

We also considered models where the generator is learned blindly at test time, *i.e.*, we learn $\boldsymbol{\theta}$ while reconstructing $\boldsymbol{x}$. While the original Deep Image Prior work left the input $\boldsymbol{z}$ fixed, here we also learn the input $\boldsymbol{z}_j$ to the generator for each patch. Without learning $\boldsymbol{z}_j$ it is unlikely that a single generator could fit all patches from random inputs. Thus, our

cost minimizes over $\boldsymbol{x}$, $\boldsymbol{z}_j$ and $\boldsymbol{\theta}$

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \min_{\boldsymbol{\theta}} \min_{\boldsymbol{z}} \frac{\lambda}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \sum_j \frac{1}{2}\|G(\boldsymbol{z}_j;\boldsymbol{\theta}) - \boldsymbol{P}_j\boldsymbol{H}\boldsymbol{x}\|_2^2. \tag{6.14}$$

$\boldsymbol{H}$ is an optional prefilter for the image, allowing the generator to fit just certain features, like edges for example. Such filtering is typical in convolutional dictionary learning methods [17]. Whereas DIP learned from a single example image, our patchwise model learns from a collection of patches from a single image. We hope that also learning $\boldsymbol{z}_j$ alleviates this greater representation requirement.

Another variation we considered was to have the latent patches $\boldsymbol{z}_j$ resemble the image patches. In this case the generator is blindly denoising the patches as follows

$$\hat{x} = \arg\min_{\boldsymbol{x}} \min_{\boldsymbol{\theta}} \min_{\boldsymbol{z}} \frac{\lambda}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \sum_j \frac{\beta}{2}\left\|G(\boldsymbol{P}_j^z\boldsymbol{H}\boldsymbol{z};\boldsymbol{\theta}) - \boldsymbol{P}_j^x\boldsymbol{H}\boldsymbol{x}\right\|_2^2 + \frac{\gamma}{2}\|\boldsymbol{z} - \boldsymbol{x}\|_2^2. \tag{6.15}$$

Here, instead of learning a $\boldsymbol{z}_j$ for every patch, we only learn one $\boldsymbol{z}$ the size of $\boldsymbol{x}$ from which we extract patches. G is constrained architecturally from learning the identity function, specifically, by using an autoencoder or hourglass architecture.

6.4 Experiments

6.4.1 Pretrained Generator Method

Following the experiments presented in Bora *et al.* [12], we start with a compressed sensing problem on MNIST digits [49]. We used patches of MNIST digits to train a Deep Convolutional Generative Adversarial Network (DCGAN) [63] to learn a generator of patches of digits using standard supervised adversarial training with a discriminator. Each digit is 28×28, resulting in an optimization vector of length 784. We compute 100 random Gaussian measurements from test MNIST images not used in training and regularize with 12×12 overlapping patches.

As an initial test, we attempt to fit the regularizer alone by minimizing $\mathsf{R}(\boldsymbol{x}_{\text{true}})$ w.r.t. $\{\boldsymbol{z}_j\}$ with $p = 2$. **FIG** 6.1 describes the process. Most generated patches are able to fit to the oracle patches, after 5 random restarts on each patch. While a few patches still exhibit some error, the artifacts are averaged out in the net affect of the regularization.

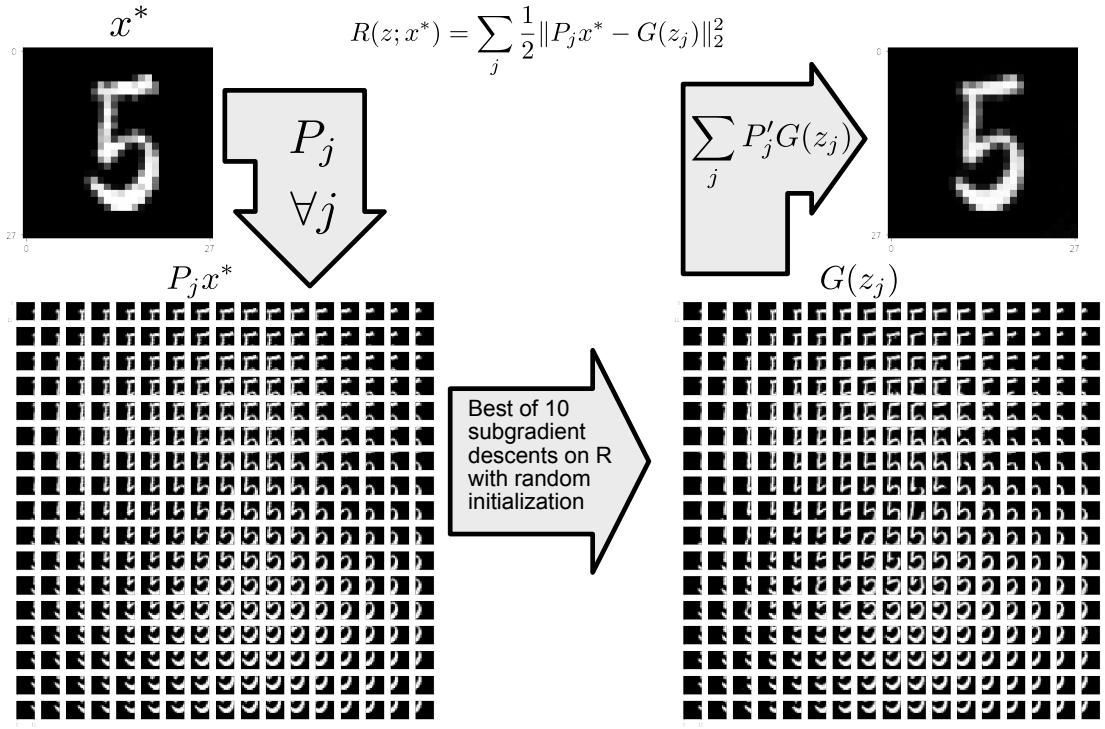


FIG 6.1 – Attempt to fit an oracle with a patch-wise generator network

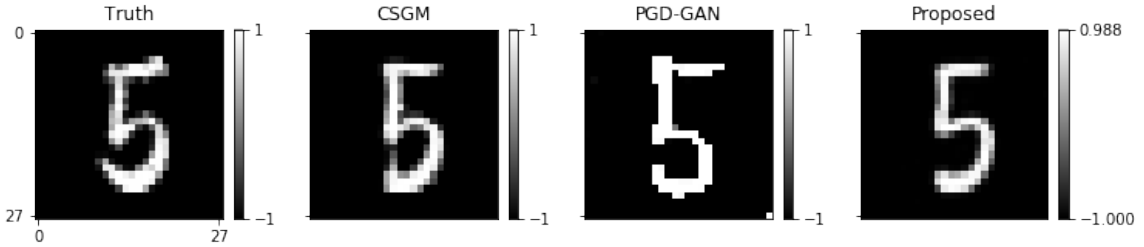


FIG 6.2 – Attempt to use a patch-wise generator in a compressed sensing reconstruction. CSGM is the method of Bora *et al.* [12] that is similar to the proposed method without being patch-wise. PGD-GAN is the method of Shah and Hegde [72]. For the proposed method, the image was initialized with the output of CSGM.

FIG 6.2 presents the results of attempting to use R in a compressed sensing reconstruction. While $\boldsymbol{z}$ does not yet find as accurate of a minimizer as CSGM with random initializations, the patch separability of the proposed method allows iterative refinement of the estimate image by random reinitializing patches and keeping the best performers. This means while CSGM generally has the best final estimate, its average estimate is much worse than the proposed method.

When we applied the proposed method in a reconstruction problem using the described BCD approach, the first iteration severely degraded the quality of the image estimate, though the following iterations slowly recovered some of that loss. This seems to be because the generator does not fully fit our initial image estimate in the first iteration and then the $\boldsymbol{x}$ update transfers that low-quality fitting to the image. To remedy this, we allowed the $\boldsymbol{z}$ update to run longer in the first iteration to fully fit the initialization and allowed for multiple restarts to help $\boldsymbol{z}$ find a good minimizer. But when we do this, the algorithm became fixed at this iterate and no further improvements beyond the initialization were made.

One idea would be to move away from a BCD approach and estimate $\{\boldsymbol{z}_j\}$ and $\boldsymbol{x}$ jointly. While this would likely be harder to optimize than even $\{\boldsymbol{z}_j\}$ is alone, we could add a stochastic element to this update by only updating a subset of $\boldsymbol{z}_j$ and patches of $\boldsymbol{x}$. Maybe this randomness could improve the optimization performance as it has been shown to do in deep neural networks.

6.4.2 Blind Generator Methods

In the previous section, I worked with MNIST data as its small size allowed for quick training and experimentation. In the blind generator methods, no pretraining is required, and so I opted to work on a more realistic problem, namely phase-encode single coil MRI reconstruction.

In the case of single coil MRI, patch extraction with wrapping end conditions, and a circulant prefilter $\boldsymbol{H}$, the $\boldsymbol{x}$ update can be computed in closed form as

$$\boldsymbol{F}^{\text{-}1}\left(\frac{\lambda \boldsymbol{M}\boldsymbol{y} + \tilde{\boldsymbol{H}}'\boldsymbol{F}\sum_j \boldsymbol{P}_j' G(\boldsymbol{z}_j;\boldsymbol{\theta})}{\lambda \boldsymbol{M}^2 + \omega\,\mathsf{diag}^2\left\{\tilde{\boldsymbol{H}}\right\}}\right), \tag{6.16}$$

where $\boldsymbol{F}$ is the Fourier transform, $\boldsymbol{M}$ is a diagonal matrix of Fourier sampling locations, $\boldsymbol{y}$ is zero-filled k-space measurements, and $\tilde{\boldsymbol{H}}$ is a diagonal matrix such that $\boldsymbol{F}^{\text{-}1}\tilde{\boldsymbol{H}}\boldsymbol{F} = \boldsymbol{H}$.

For a prefilter, we used the Tikhonov high pass filter that has been used in dictionary methods to only learn edges

$$\boldsymbol{H} = \boldsymbol{I} - (\boldsymbol{I} + \lambda_{\boldsymbol{H}} \boldsymbol{C}'\boldsymbol{C})^{-1}, \tag{6.17}$$

where $\boldsymbol{C}$ is a (2D) finite-difference operator. In our experiments, we also tried $\boldsymbol{H} = \boldsymbol{I}$.

In our experiments, both blind methods, with or without $\boldsymbol{H}$, performed little better than a zero-fill reconstruction, even if the initialization was much better. Generally, whatever the generator could learn to represent in one iteration was where the algorithm stayed for any future iteration. Thus, if we increased the number of inner $\boldsymbol{\theta}$ iterations in the first iterations, we could fit our better initialization, but then the algorithm would be fixed there.

We also found that the original Deep Decoder and Deep Image Prior methods did not perform as well as standard sparsity based methods such as UTL or SOUP for this single-coil MRI application on the images we were working with.

6.5 Conclusion

Despite trying many variations on these algorithms, we were unable to find any potential for significant gains. The non-convexity of neural networks makes them difficult to optimize on a single set of patches. Because there were so many patches we were optimizing over, it was common to see a large portion of them get stuck in poor local minima while optimizing. Compared to restarting the optimization of a single image, detecting poor patches and restarting them is more challenging. For this reason, these models do not seem as attractive for adaptive regularization compared to other methods with more efficient updates.

CHAPTER 7

Dynamic Subspace Estimation with Piecewise Geodesics

Dynamic subspace estimation, or subspace tracking, is a fundamental problem in statistical signal processing and machine learning. This chapter considers a piecewise geodesic model for time-varying subspaces. The natural objective function for this model is non-convex. We propose a novel algorithm for minimizing this objective and estimating the parameters of the model from data with Grassmannian-constrained optimization. We show that with this algorithm, the objective is monotonically non-increasing. We demonstrate the performance of this model and our algorithm on synthetic data, video data, and dynamic fMRI data.

7.1 Introduction

Modeling data using linear subspaces is a powerful analytical tool that enables practitioners to more efficiently and reliably solve high-level tasks like inference and decision making, classification, and anomaly detection, among others. In some applications of interest, the data generation process is time-varying or dynamic in nature, which motivates the use of a dynamic linear subspace for data modeling. Some example applications where dynamic subspace models are prevalent include array signal processing [90, 28, 74, 47], communication systems [35], video processing [85], and dynamic magnetic resonance imaging (MRI) [61]. The goal in these applications is to learn a time-varying subspace from the observed data.

Most previous theoretical work for modeling a dynamic subspace relies on very strong assumptions of the dynamics – either assuming very simple dynamics like sudden changes

This chapter based on work submitted for double-blinded review and was written in collaboration with Haroon Raja, Jeffrey A. Fessler, and Laura Balzano. Experiments on video data and associated figures were done by Haroon Raja.

with an otherwise static subspace, or assuming a specific known dynamical model. A much broader empirical literature for *subspace tracking* considers a wide range of algorithms with different strengths and weaknesses with regards to signal-to-noise ratios, speed of dynamics, and computational complexity. For the vast majority of these algorithms, accuracy guarantees in the presence of dynamics are still an open question.

The first contribution of this paper is to study a flexible and natural dynamic subspace model: the piecewise geodesic model. A piecewise geodesic can approximate any curve on the Grassmannian, *i.e.*, any continuously varying subspace. This model generalizes both the previously studied time-varying subspace models and piecewise linear approximations that are pervasive in the theory and practice of statistical signal processing. This model has only been very briefly discussed in existing literature, probably in part due to the difficulty of parameter estimation and algorithmic guarantees in this setting. The central contribution of this paper, therefore, is an algorithm for learning the parameters of this model in a batch setting that is guaranteed to descend an appropriate cost function (corresponding to a log-likelihood for Gaussian noise) at every step. We also demonstrate the performance of the proposed algorithm empirically on both synthetic and real datasets.

7.1.1 Problem Formulation and Geodesic Model

We start with the following broad generative model for data arising from a time-varying subspace. At each time point i we observe ℓ vectors from a time-varying subspace. Let $\boldsymbol{X}_i \in \mathbb{R}^{d\times\ell}$ for $i = 1, 2, \ldots, T$ be data generated from a low-rank model with noise:

$$\boldsymbol{X}_i = \boldsymbol{U}_i\boldsymbol{G}_i + \boldsymbol{N}_i \tag{7.1}$$

where $\boldsymbol{U}_i \in \mathbb{R}^{d\times k}$ is a matrix with orthonormal columns representing a point on the Grassmannian $\mathcal{G}(k, d)$, the space of all rank-k subspaces in $\mathbb{R}^d$; $\boldsymbol{G}_i \in \mathbb{R}^{k\times\ell}$ holds weight or loading vectors; and $\boldsymbol{N}_i \in \mathbb{R}^{d\times\ell}$ is an independent additive noise matrix. We observe $\boldsymbol{X}_i$ and our objective is to estimate $\boldsymbol{U}_i$ for $i = 1, \ldots, T$. Note that while we use "time-varying" to describe this generative model, in practice "i" could represent some dimension other than time, and the algorithms we consider are batch in the sense that they use all the data $\boldsymbol{X}_i$, $i = 1, \ldots, T$ for estimating $\boldsymbol{U}_i$, $i = 1, \ldots, T$.

If $\boldsymbol{U}_i = \bar{\boldsymbol{U}}$ is static for all $i = 1, 2, \ldots, T$, and if $\boldsymbol{G}_i$ has zero-mean columns, then we could concatenate all $\boldsymbol{X}_i$ together and apply the SVD, which is well-known to recover a good approximation of $\bar{\boldsymbol{U}}$ as long as the number of samples ℓT is large enough to overcome the noise. However, if $\boldsymbol{U}_i$ is varying for every i, one can immediately see that if $\ell < k$,

estimating $\boldsymbol{U}_i$ is impossible without further assumptions. Even when $\ell \geq k$, in many applications it is natural to impose a relationship between the $\boldsymbol{U}_i$ subspace matrices over time, to guarantee regularity properties or known application constraints. Various constraints have been studied in the literature, such as a slowly rotating subspace, a subspace that is mostly static except for intermittent sudden changes, or a subspace that changes one dimension at a time [57]. Those models are all subsumed by the **piecewise geodesic** model for dynamic subspaces, illustrated in FIG 7.1. In this work, we focus on efficiently learning a single geodesic.

Model for a Single Geodesic Let $2k \leq d$. We model each $\boldsymbol{U}_i$ as an orthonormal basis whose span has been sampled from a single continuous Grassmannian geodesic $\boldsymbol{U}(t) : [0, 1] \to \mathcal{V}^{d\times k}$ parameterized as follows:

$$\boldsymbol{U}_i = \boldsymbol{U}(t_i) = \boldsymbol{H}\cos(\boldsymbol{\Theta}t_i) + \boldsymbol{Y}\sin(\boldsymbol{\Theta}t_i) \tag{7.2}$$

where $\mathcal{V}^{d\times k}$ is the set of $d \times k$ matrices with orthonormal columns (the Stiefel manifold), $\boldsymbol{H} \in \mathcal{V}^{d\times k}$ is an orthonormal basis for a point on the Grassmannian, $\boldsymbol{Y}$ is a matrix with orthonormal columns whose span is in the tangent space of the Grassmannian at $\mathrm{span}(\boldsymbol{H})$, *i.e.*, $\boldsymbol{Y} \in \left\{\boldsymbol{Y} | \boldsymbol{H}^\top \boldsymbol{Y} = \boldsymbol{0}, \boldsymbol{Y} \in \mathcal{V}^{d\times k}\right\}$, and $\boldsymbol{\Theta} \in \mathbb{R}^{k\times k}$ is a diagonal matrix where θ_j is the jth principal angle between the two endpoints of the geodesic, and sine/cosine are the matrix versions. These constraints ensure each $\boldsymbol{U}_i$ has orthonormal columns. The scalars $t_i \in [0, 1]$ represent the location of each $\boldsymbol{U}_i$ along the geodesic, *i.e.*, time-points scaled (or normalized) to the interval if the geodesic is sampled over time. For more information, see [1, Section 3.8] and [24].

Because we are only interested in the span of $\boldsymbol{U}_i$, this parameterization of a Grassmannian geodesic is not unique. Permuting the columns of $\boldsymbol{H}$, $\boldsymbol{Y}$ and the diagonal elements of $\boldsymbol{\Theta}$ would result in a $\boldsymbol{U}_i$ with the same span. Additionally, there is a sign ambiguity between columns of $\boldsymbol{Y}$ and diagonal elements of $\boldsymbol{\Theta}$. In practice, our loss is invariant to these ambiguities and so they are not a problem. Any specific parameterization can easily be transformed into another.

$\boldsymbol{H}$, $\boldsymbol{Y}$ and $\boldsymbol{\Theta}$ are all learnable parameters describing $\boldsymbol{U}(t)$. Conceptually, we can think of $\boldsymbol{H}$ as a starting point on the Grassmannian, $\boldsymbol{Y}$ as a normalized direction we want to walk, and the products $\boldsymbol{\Theta}t_i$ as the distances in each dimension we should walk from $\boldsymbol{H}$ on the surface of the manifold to get to $\boldsymbol{U}_i$.

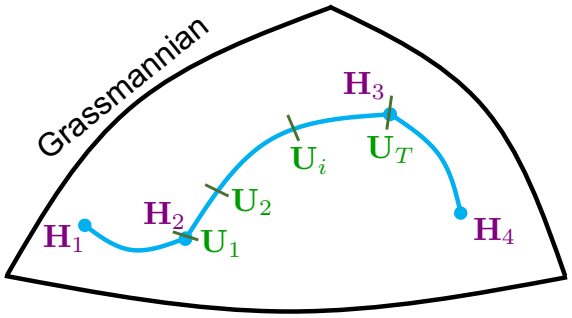


FIG 7.1 – Illustration of the piecewise geodesic model. $\mathbf{H_1}, \ldots, \mathbf{H_4}$ are points on the Grassmannian. When estimating a single geodesic, *e.g.*, the one from $\mathbf{H_2}$ to $\mathbf{H_3}$, then $\boldsymbol{H}$ is an orthonormal basis for $\mathbf{H_2}$ and $\boldsymbol{Y}$ is an orthonormal basis for $(\boldsymbol{I} - \boldsymbol{H}\boldsymbol{H}')\mathbf{H_3}$.

Single Geodesic *vs.* Piecewise Geodesic In this work, we focus on learning a single geodesic from data with given time points t_i. This focus essentially makes two key simplifying assumptions: (1) the locations of the knots, or change-points, in a piecewise approximation are given, and (2) between two knots in the piecewise approximation, either the time-points t_i are given, or observed matrices $\boldsymbol{X}_i$ are equidistant along a geodesic curve. With these assumptions, our high-level approach is to take each set of data matrices between change-points and learn a single geodesic. We plan to relax both of these assumptions in future work, see Section 7.4.

7.1.2 Related work

Classical literature on subspace tracking uses online approaches to estimate the time-varying subspaces [90, 16, 5, 7, 35, 57, 85, 19]. Early theoretical results were limited to asymptotic convergence guarantees with static underlying subspaces. Among the more recent works, the PETRELS algorithm [16] portrays a recursive least squares approach and provides convergence theory that assumes that the subspace changes at a particular instant and then stays constant for sufficient time so that the change can be tracked (also called the piecewise constant model). Narayanamurthy and Vaswani [57] relax the assumption of constant subspace to a very slowly varying subspace between the change points. For a review of these methods, see [85].

Dynamic subspace estimation has also been studied for the more general Grassmannian geodesic model [47, 28, 74, 39], also the focus of this paper. Unlike the subspace tracking problem, these contributions focus more on batch data settings with access to the whole dataset for estimation, which is the approach we take herein. For example,

Lake and Keenan [47] formulate the subspace tracking for any given epoch in the generative geodesic model in the form of a likelihood function to maximize. The likelihood is non-concave, but the authors provide an annealing approach to solve it, which will have very high computational burden for many modern high-dimensional and large-data applications. Fuhrmann [28] and Srivastava and Klassen [74] have also studied the generative geodesic model. The solution provided by Srivastava and Klassen [74] is not applicable to large-scale settings since it relies on a sampling based strategy like Markov chain Monte-Carlo (MCMC) and hence is computationally intensive. On the other hand, the solution provided by Fuhrmann [28] is computationally inexpensive, but it only handles one-dimensional subspaces ($k = 1$). In summary, major weaknesses of the state-of-the-art methods include high computational costs, lack of theoretical guarantees, and/or need to tune hyperparameters. This chapter approaches the problem using modern non-convex optimization tools, alleviating these issues by devising an algorithm that is parameter free other than the subspace dimension, descends the loss function monotonically, and uses thin SVDs to solve the problem.

7.2 Method

Given data and associated time points $\{\boldsymbol{X}_i, t_i\}_{i=1}^T$, we fit the proposed geodesic model for $\boldsymbol{U}(t)$ by minimizing the following loss function

$$\mathcal{L}(\boldsymbol{U}) = \mathcal{L}(\boldsymbol{H}, \boldsymbol{Y}, \boldsymbol{\Theta}) = \min_{\{\boldsymbol{G}_i\}_{i=1}^T} \sum_{i=1}^T \|\boldsymbol{X}_i - \boldsymbol{U}(t_i)\boldsymbol{G}_i\|_{\mathrm{F}}^2 \tag{7.3}$$

$$= -\sum_{i=1}^T \|\boldsymbol{X}_i' \boldsymbol{U}(t_i)\|_{\mathrm{F}}^2 + c, \tag{7.4}$$

where in the last equality we have substituted the optimal $\boldsymbol{G}_i = (\boldsymbol{U}(t_i))'\boldsymbol{X}_i$ and simplified and c is a constant [31]. At a high level, we minimize our loss function with respect to $\boldsymbol{H}$, $\boldsymbol{Y}$ and $\boldsymbol{\Theta}$ via block coordinate descent. Our first block updates $(\boldsymbol{H}, \boldsymbol{Y})$ jointly via an SVD of a $d \times 2k$ matrix. Next, we update $\boldsymbol{\Theta}$ via a first-order iterative minimization. Specifically, we design an efficient majorize-minimize (MM) iteration for updating $\boldsymbol{\Theta}$.

7.2.1 $(\boldsymbol{H}, \boldsymbol{Y})$ Update

Let $\boldsymbol{Q} \triangleq [\boldsymbol{H}\ \boldsymbol{Y}]$ and $\boldsymbol{Z}_i \triangleq [\cos(\boldsymbol{\Theta} t_i);\ \sin(\boldsymbol{\Theta} t_i)]$. Then we rewrite our model (7.2) as $\boldsymbol{U}_i = \boldsymbol{Q}\boldsymbol{Z}_i$. By constraining $\boldsymbol{Q} \in \mathcal{V}^{d\times 2k}$, we also satisfy the constraints of $\boldsymbol{H}$ and $\boldsymbol{Y}$ individually.

Following [14], we form a linear majorizer for our loss (7.4) at $\boldsymbol{Q}^{(n)}$ and minimize it with a Stiefel constraint simply by projecting its negative gradient onto the Stiefel manifold. This projection is just the solution to the generalized Procrustes problem (*c.f.* Section 2.4). The update is then given by

$$\boldsymbol{Q}^{(n+1)} = \underset{\boldsymbol{Q}\in\mathcal{V}^{d\times 2k}}{\arg\min} \left\| \left(\sum\nolimits_{i=1}^{T} \boldsymbol{X}_i \boldsymbol{X}_i' \boldsymbol{Q}^{(n)} \boldsymbol{Z}_i \boldsymbol{Z}_i' \right) - \boldsymbol{Q} \right\|_{\mathrm{F}}^2 \tag{7.5}$$

$$= \boldsymbol{W}\boldsymbol{V}', \text{ where} \tag{7.6}$$

$$\boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{V}' \text{ is the SVD of } \sum_{i=1}^{T} \boldsymbol{X}_i \boldsymbol{X}_i' \boldsymbol{Q}^{(n)} \boldsymbol{Z}_i \boldsymbol{Z}_i' \tag{7.7}$$

$$= \sum_{i=1}^{T} \left[\boldsymbol{X}_i \hat{\boldsymbol{G}}_i{}' \cos(\boldsymbol{\Theta} t_i) \quad \boldsymbol{X}_i \hat{\boldsymbol{G}}_i{}' \sin(\boldsymbol{\Theta} t_i) \right],$$

where in the last line we let $\hat{\boldsymbol{G}}_i' = \boldsymbol{X}_i' \boldsymbol{U}_i^{(n)} = \boldsymbol{X}_i' \boldsymbol{Q}^{(n)} \boldsymbol{Z}_i$. Although here we eliminated $\{\boldsymbol{G}_i\}_{i=1}^T$ from our loss (7.4), if we had not, this same update could be derived as a block coordinate update on $\boldsymbol{Q}$ and $\{\boldsymbol{G}_i\}_{i=1}^T$, where, again, the $\boldsymbol{Q}$ update would be a generalized Procrustes problem. See Appendix B.1.1 for more details.

7.2.2 $\boldsymbol{\Theta}$ Update

The loss (7.4) for fixed $\boldsymbol{H}$ and $\boldsymbol{Y}$ is a smooth function of $\boldsymbol{\Theta}$ and can be effectively optimized via an iterative quadratic majorize-minimize scheme. Here we provide an overview of the method. Appendix B.1.2 provides a more complete derivation of the majorizer. First, we simplify the loss (7.4)

$$\hat{\boldsymbol{\Theta}} = \underset{\boldsymbol{\Theta}}{\arg\min} - \sum_{i=1}^{T} \left\| \boldsymbol{X}_i' \boldsymbol{U}_i \right\|_{\mathrm{F}}^2 \tag{7.8}$$

$$= \underset{\boldsymbol{\Theta}}{\arg\min} - \sum_{i=1}^{T} \left\| \boldsymbol{X}_i' \left(\boldsymbol{H} \cos(\boldsymbol{\Theta} t_i) + \boldsymbol{Y} \sin(\boldsymbol{\Theta} t_i) \right) \right\|_{\mathrm{F}}^2 \tag{7.9}$$

$$= \underset{\boldsymbol{\Theta}}{\arg\min} - \sum_{i=1}^{T} \sum_{j=1}^{k} r_{i,j} \cos(2\theta_j t_i - \phi_{i,j}) + b_{i,j}, \tag{7.10}$$

where defining $\operatorname{arctan2}(y, x)$ as the angle of the point (x, y) in the 2D plane counter-clockwise from the positive x-axis, the associated constants $r_{i,j}, \phi_{i,j}, b_{i,j}$ are defined as

$$\begin{aligned} \phi_{i,j} &= \operatorname{arctan2}\left(\beta_{i,j}, \frac{\alpha_{i,j} - \gamma_{i,j}}{2}\right) \\ r_{i,j} &= \sqrt{\left(\frac{\alpha_{i,j} - \gamma_{i,j}}{2}\right)^2 + \beta_{i,j}^2} \\ b_{i,j} &= \frac{\alpha_{i,j} + \gamma_{i,j}}{2} \end{aligned} \qquad \begin{aligned} \alpha_{i,j} &= [\boldsymbol{H}'\boldsymbol{X}_i\boldsymbol{X}_i'\boldsymbol{H}]_{j,j} \\ \beta_{i,j} &= \operatorname{real}\Big\{[\boldsymbol{Y}'\boldsymbol{X}_i\boldsymbol{X}_i'\boldsymbol{H}]_{j,j}\Big\}. \\ \gamma_{i,j} &= [\boldsymbol{Y}'\boldsymbol{X}_i\boldsymbol{X}_i'\boldsymbol{Y}]_{j,j} \end{aligned} \tag{7.11}$$

This loss is separable for each diagonal element of $\boldsymbol{\Theta}$, so we find each $\hat{\theta}_j$ via a (1D) minimization. Let $f_{i,j}(\theta_j) \triangleq -r_{i,j}\cos(2\theta_j t_i - \phi_{i,j}) + b_{i,j}$. Then (*c.f.* [29]) the following $q_{i,j}$ defines a quadratic majorizer for $f_{i,j}$ at θ_j

$$q_{i,j}\big(\theta_j; \bar{\theta}_j\big) = f_{i,j}\big(\bar{\theta}_j\big) + \dot{f}_{i,j}\big(\bar{\theta}_j\big)(\theta_j - \bar{\theta}_j) + \frac{1}{2}\, w_{f_{i,j}}\big(\bar{\theta}_j\big)\,(\theta_j - \bar{\theta}_j)^2 \tag{7.12}$$

$$\geq f_{i,j}(\theta_j) \tag{7.13}$$

where the derivative $\dot{f}_{i,j}$ and curvature function $w_{f_{i,j}}$ are given by

$$\dot{f}_{i,j}(\theta_j) = 2r_{i,j}t_i \sin(2\theta_j t_i - \phi_{i,j}) \qquad \text{and} \tag{7.14}$$

$$w_{f_{i,j}}(\theta_j) = \frac{\dot{f}_{i,j}(\theta_j)}{\operatorname{mod}\left((\theta_j - \frac{\phi_{i,j}}{2t_i}) + \frac{\pi}{2t_i}, \frac{2\pi}{2t_i}\right) - \frac{\pi}{2t_i}}. \tag{7.15}$$

Appendix B.1.2.2 gives a detailed construction of $w_{f_{i,j}}$.

Our majorize-minimize iterations for each diagonal element of $\boldsymbol{\Theta}$ are then given by

$$\theta_j^{(n+1)} = \arg\min_{\theta_j} \sum_{i=1}^{T} q_{i,j}\big(\theta_j; \theta_j^{(n)}\big) \tag{7.16}$$

$$= \theta_j^{(n)} - \frac{\sum_{i=1}^{T} \dot{f}_{i,j}\big(\theta_j^{(n)}\big)}{\sum_{i=1}^{T} w_{f_{i,j}}\big(\theta_j^{(n)}\big)}. \tag{7.17}$$

Conceptually, each MM update can be interpreted as a gradient descent step with a variable step size $s^{(n)} = 1/(\sum_{i=1}^{T} w_{f_{i,j}}\big(\theta_j^{(n)}\big))$ that is guaranteed to not increase the loss even without any line search. Indeed, as both updates just outlined are known for their monotonicity properties, we have the following monotonicity result for our overall algorithm. In practice, we see global convergence in the vast majority of experiments, but not

in all. A more thorough investigation of the loss landscape and algorithmic convergence properties is of great interest for future work.

Theorem 1 *Algorithm 2 produces iterates* $\boldsymbol{U}^{(n)}(t) = \boldsymbol{H}^{(n)} \cos(\boldsymbol{\Theta}^{(n)} t) + \boldsymbol{Y}^{(n)} \sin(\boldsymbol{\Theta}^{(n)} t)$ *that are monotonically non-increasing in loss (7.4), i.e.,* $\mathcal{L}(\boldsymbol{U}^{(n+1)}) \leq \mathcal{L}(\boldsymbol{U}^{(n)})$.

Proof: It suffices to show that each block coordinate update does not increase the loss. Both the $[\boldsymbol{H}\ \boldsymbol{Y}]$ block and $\boldsymbol{\Theta}$ block updates are instances of MM methods that guarantee this property. See, for example, Sun *et al.* [76, Section II.C] for a general treatment and Breloy *et al.* [14, Section III.B] for MM convergence with the non-convex Stiefel constraint.

7.3 Experiments

To show the effectiveness of the proposed method, we present results on both synthetic and real data. On the synthetic data, we show the effect of the different data parameters, such as d, k, ℓ, T and the additive noise standard deviation σ. With the intuition we build on the synthetic data, we present results on real measured data and show how we determine the underlying rank. All experiments were performed on a 2021 Macbook Pro laptop computer and implemented in Python.

7.3.1 Synthetic Data

In the case of synthetic data, we are able to compare our estimated geodesic $\hat{\boldsymbol{U}}(t)$ against the true geodesic from which the data was generated. Our error metric is the square root of the average squared subspace error between corresponding points along the geodesic

$$\text{Subspace Error} = \frac{1}{\sqrt{2k}} \left\| \hat{\boldsymbol{U}}\hat{\boldsymbol{U}}' - \boldsymbol{U}\boldsymbol{U}' \right\|_{\mathrm{F}} \tag{7.18}$$

$$\text{Geodesic Error} = \sqrt{\int_0^1 \frac{1}{2k} \left\| \hat{\boldsymbol{U}}(t)\hat{\boldsymbol{U}}(t)' - \boldsymbol{U}(t)\boldsymbol{U}(t)' \right\|_{\mathrm{F}}^2 \mathrm{d}t} \,. \tag{7.19}$$

In practice, we approximate the integral by sampling the geodesic at a large number of time points. The subspace error (7.18) takes a minimum value of 0 when $\operatorname{span}(\hat{\boldsymbol{U}}) = \operatorname{span}(\boldsymbol{U})$ and a maximum value of 1 when $\operatorname{span}(\hat{\boldsymbol{U}}) \perp \operatorname{span}(\boldsymbol{U})$. As such, the geodesic error (7.19) is similarly bounded. For all of our synthetic experiments, we generated data from our planted model (7.1). $\boldsymbol{\Theta}$ was constrained to generate distance-minimizing geodesics,

Algorithm 2 Geodesic Subspace Estimation

Require: $\{\boldsymbol{X}_i, t_i\}_{i=1}^{T}$, $\boldsymbol{H}^{(0)}, \boldsymbol{Y}^{(0)}, \boldsymbol{\Theta}^{(0)}$, N = # of outer iterations, M = # of inner MM iterations

for $n = 1, \ldots, N$ **do**

 # $\boldsymbol{H}, \boldsymbol{Y}$ update

$$\boldsymbol{U}_i^{(n-1)} = \boldsymbol{H}^{(n-1)} \cos\left(\boldsymbol{\Theta}^{(n-1)} t_i\right) + \boldsymbol{Y}^{(n-1)} \sin\left(\boldsymbol{\Theta}^{(n-1)} t_i\right) \quad \forall i$$

$$\boldsymbol{G}_i = (\boldsymbol{U}_i^{(n-1)})' \boldsymbol{X}_i \quad \forall i$$

$$\boldsymbol{M} = \sum_{i=1}^{T} \left[\boldsymbol{X}_i \boldsymbol{G}_i' \cos\left(\boldsymbol{\Theta}^{(n-1)} t_i\right) \quad \boldsymbol{X}_i \boldsymbol{G}_i' \sin\left(\boldsymbol{\Theta}^{(n-1)} t_i\right)\right]$$

$$\boldsymbol{W}, \boldsymbol{\Sigma}, \boldsymbol{V}' = \text{SVD}(\boldsymbol{M})$$

$$[\boldsymbol{H}^{(n)} \; \boldsymbol{Y}^{(n)}] = \boldsymbol{W}\boldsymbol{V}'$$

 # $\boldsymbol{\Theta}$ update

$$\boldsymbol{\Theta}^{(n,0)} = \boldsymbol{\Theta}^{(n-1)}$$

for $j = 1, \ldots, k$ **do**

$$\alpha_{i,j} = [\boldsymbol{H}' \boldsymbol{X}_i \boldsymbol{X}_i' \boldsymbol{H}]_{j,j} \quad \forall i$$

$$\beta_{i,j} = \text{real}\left\{[\boldsymbol{Y}' \boldsymbol{X}_i \boldsymbol{X}_i' \boldsymbol{H}]_{j,j}\right\} \quad \forall i$$

$$\gamma_{i,j} = [\boldsymbol{Y}' \boldsymbol{X}_i \boldsymbol{X}_i' \boldsymbol{Y}]_{j,j} \quad \forall i$$

$$\phi_{i,j} = \arctan2\left(\beta_{i,j}, \frac{\alpha_{i,j} - \gamma_{i,j}}{2}\right) \quad \forall i$$

$$r_{i,j} = \sqrt{\left(\frac{\alpha_{i,j} - \gamma_{i,j}}{2}\right)^2 + \beta_{i,j}^2} \quad \forall i$$

for $m = 1, \ldots, M$ **do**

$$z_j = \sum_{i=1}^{T} 2 r_{i,j} t_i \sin\left(2\theta^{(n,m-1)}{}_j t_i - \phi_{i,j}\right)$$

$$w_j = \sum_{i=1}^{T} \frac{2 r_{i,j} t_i \sin(2\theta^{(n,m-1)}{}_j t_i - \phi_{i,j})}{\text{mod}\left((\theta^{(n,m-1)}{}_j - \frac{\phi_{i,j}}{2t_i}) + \frac{\pi}{2t_i}, \frac{2\pi}{2t_i}\right) - \frac{\pi}{2t_i}}$$

$$\theta^{(n,m)}{}_j = \theta^{(n,m-1)}{}_j - \frac{z_j}{w_j}$$

end for

end for

$$\boldsymbol{\Theta}^{(n)} = \boldsymbol{\Theta}^{(n,M)}$$

end for

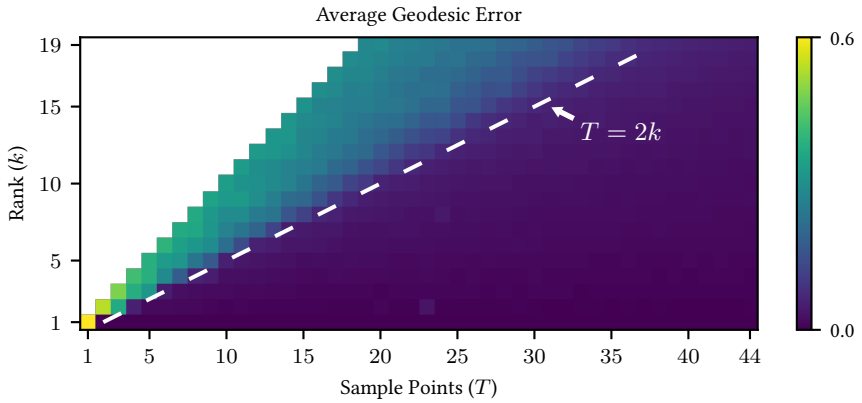


FIG 7.2 – The average geodesic error over 15 trials for varying rank k and number of sample points T. One vector was sampled at each of T points ($\ell = 1$). The ambient dimension was $d = 40$, and we added zero-mean white Gaussian noise with standard deviation $\sigma = 10^{-5}$. We see a phase transition at $T = 2k$; with at least this many samples, we recover the true subspace with low error.

i.e., $\|\boldsymbol{\Theta}\|_2 < \pi/2$. We drew $\boldsymbol{G}_i$ from a standard normal distribution. The noise $\boldsymbol{N}_i$ is additive white Gaussian noise (AWGN) with standard deviation σ. Unless otherwise noted, we initialized the proposed method with a random geodesic for all experiments. For the experiments in **FIG** 7.2 and **FIG** 7.3, we formed a course estimate of the starting and ending rank-k subspaces, and computed a geodesic between these subspaces with [1, (19)] for initialization.

FIG 7.2 shows the average geodesic error when $\ell = 1$, *i.e.*, we receive one vector per k-dimensional subspace, as a function of both the true underlying rank (k) and the number of sample points (T). This plot shows that a phase transition occurs at $T = 2k$, where if we have at least this many samples, the proposed method can recover the true geodesic with low error. Because computing the rank-$2k$ SVD requires $2k$ samples, this bound is the best we could hope to do.

In **FIG** 7.3(a), we further investigate the effect of the number of samples on the average geodesic error. Because a rank-k geodesic spans a space as large as $2k$, we have also shown for reference the subspace error of recovering a rank-$2k$ subspace with an SVD under two different distributions of loading vectors. The dashed lines show the subspace recovery error for data generated isotropically in a rank-$2k$ subspace with additive white Gaussian noise. The dotted lines show the subspace recovery error for data distributed on a geodesic in the rank-$2k$ subspace. Both SVD-based methods are only recovering a single rank-$2k$ subspace and not recovering a geodesic. Empirically, we can see that the sample

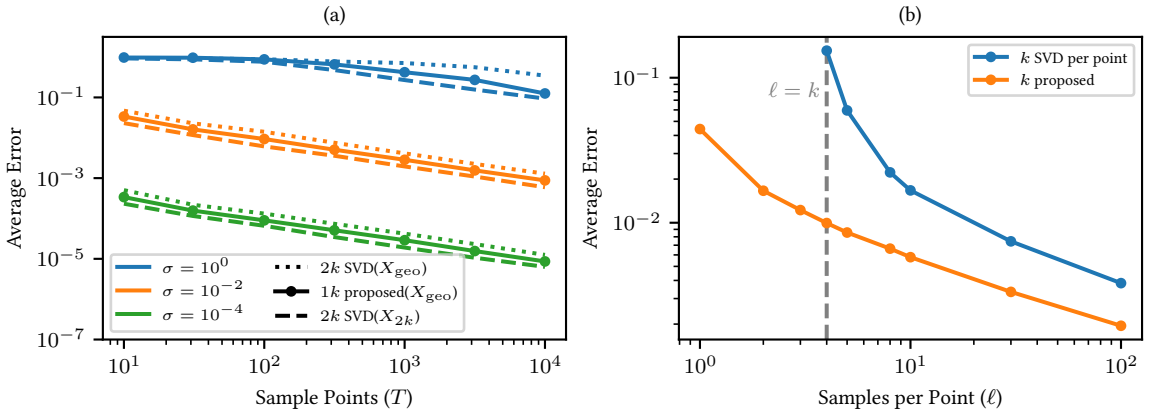


FIG 7.3 – **(a)** The recovered geodesic error (solid lines) as a function of sample size and additive noise standard deviation, averaged over 10 problems. $\boldsymbol{X}_{\text{geo}}$ represents the batched geodesic data that has data distributed like a geodesic in a rank-$2k$ subspace. $\boldsymbol{X}_{2k}$ denotes data that is distributed isotropically in a rank-$2k$ subspace. For data generated from a geodesic, the proposed method recovers the geodesic error with a lower error than an SVD can estimate its span. **(b)** Average geodesic error over 100 trials for varying number of samples (ℓ) collected at each time point for a fixed number of time points ($T = 11$) on a planted rank-4 geodesic with AWGN $\sigma = 10^{-2}$. When $\ell \geq k$ we can estimate the subspace with the SVD on just those ℓ samples. The geodesic model can estimate the subspaces even when $\ell < k$ and leverages all of the data to produce lower error.

complexity of the proposed geodesic model and method tracks well with rank-$2k$ SVD and outperforms SVD on geodesic data.

When the number of samples per time point (ℓ) is less than k the proposed method is still able to recover the true geodesic given that T is large enough. When $\ell \geq k$, one could estimate the subspace at each time point by applying a rank-k SVD at each time point. But, as shown in **FIG** 7.3(b), even when $\ell \geq k$, the proposed method recovers the subspaces at each time point with lower error for data generated from a geodesic.

Like a low-rank SVD approximation, our method requires choosing the rank k before fitting. **FIG** 7.4 shows the loss as a function of assumed rank for data generated from a rank-1 geodesic (left) and rank-2 geodesic (right). Because a rank-k geodesic spans a rank-$2k$ subspace, rank-k and rank-$2k$ SVD results are shown for comparison. Rank-$2k$ SVD will always have a lower loss by definition. Similarly, the proposed model will always have a lower loss than the rank-k SVD, since a rank-k subspace is a special case of a rank-k geodesic. Thus, we can lower-bound and upper-bound the loss of the proposed model on any data by a rank-$2k$ and rank-k SVD respectively. Additionally, if the data

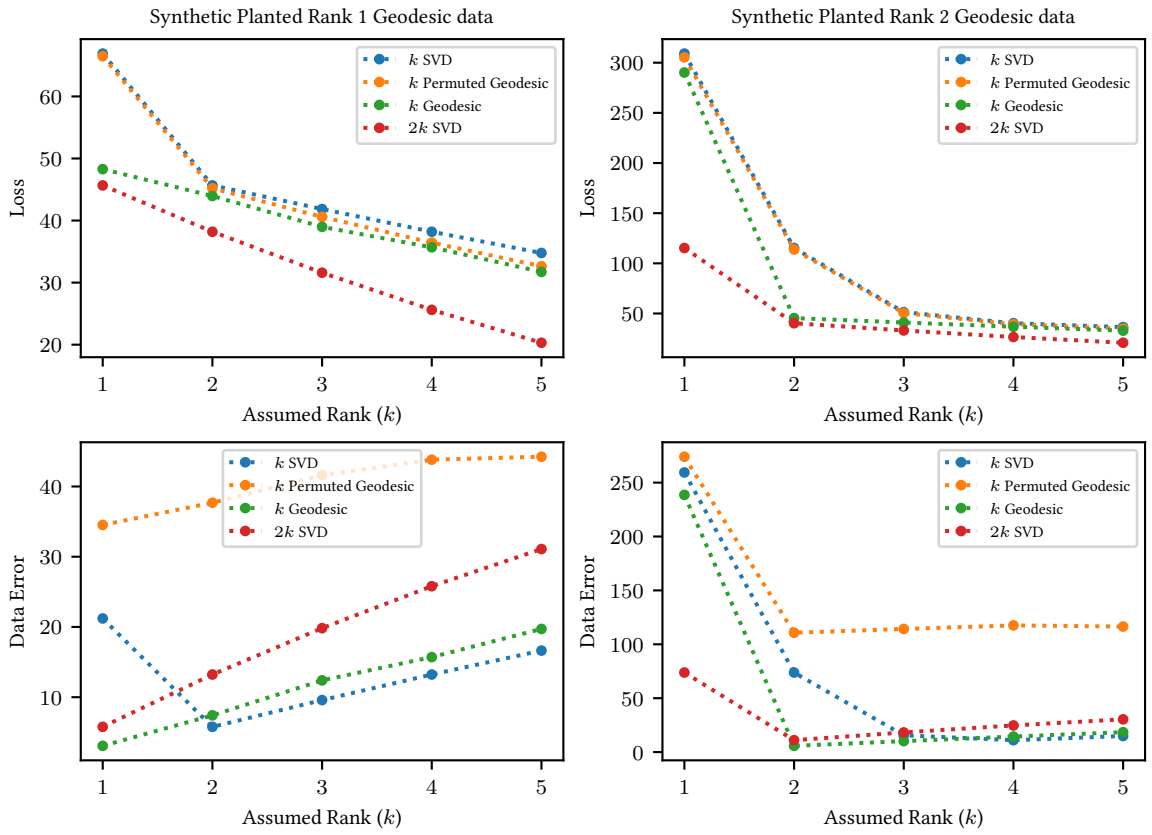


FIG 7.4 – Loss on synthetic data for a rank-1 geodesic (left) and a rank-2 geodesic (right). The loss of the proposed method is lower-bounded and upper-bounded by rank-$2k$ and rank-k SVD, respectively. When the assumed rank is equal to the true rank, than the loss of the proposed method is much closer to that of rank-$2k$ SVD, while permuting the data significantly increases the loss. From this, it is easy to deduce the true rank. The second row shows the "Data Error," which is the norm of the residual between the projected noisy data and the noiseless data. We can see that rank-$2k$ SVD was overfitting noise to obtain a lower training loss, but this increases its error. The proposed method has a lower error than rank-$2k$ SVD as long as the assumed rank is greater than or equal to the true rank on geodesic data.

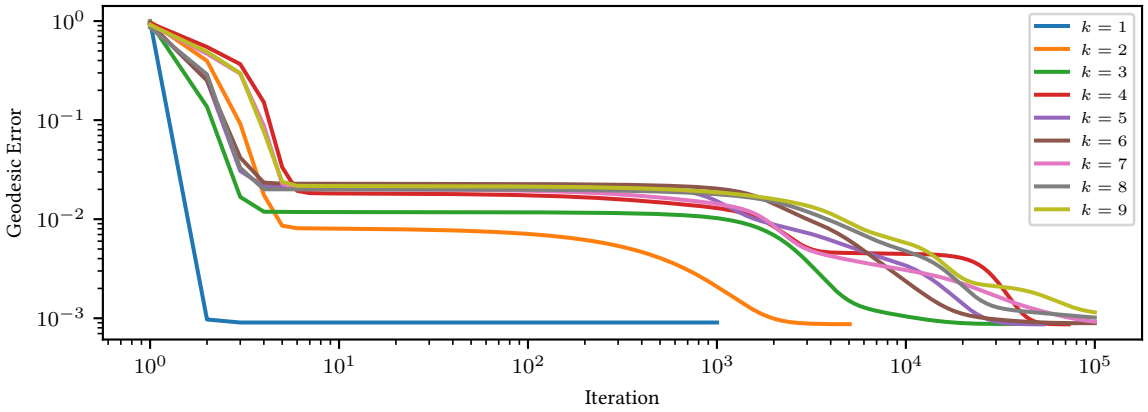


FIG 7.5 – Convergence of the proposed algorithm in geodesic error to the true geodesic for a single run of nine planted geodesic models with varying ranks in $\mathbb{R}^{40}$. The planted models were used to generate 100 sample points ($T = 100, \ell = 1$) and AWGN with standard deviation $\sigma = 10^{-3}$ was added. The algorithm was initialized with a random geodesic.

has geodesic structure, then it is ordered. For comparison we also show the loss of the proposed method on data that was generated from a geodesic and then permuted. We see that when the assumed rank is equal to the true rank of the underlying geodesic, the proposed method produces a loss much closer to that of a rank-$2k$ SVD while the proposed method applied to permuted data produces a loss much closer to that of a rank-k SVD. For permuted unordered data, the proposed model learns a geodesic with small values of $\boldsymbol{\Theta}$, approximating a static rank-k subspace similar to a rank-k SVD. For comparison, **FIG** 7.4 also shows the data error, which is the norm of the residual between the projected noisy data and the noiseless data. These plots show that rank-$2k$ SVD overfits the noise and has a higher error. The proposed method has a lower error than rank-$2k$ SVD for any assumed rank greater than or equal to the true rank on geodesic data.

FIG 7.5 shows the geodesic error per iteration of the proposed algorithm applied to several synthetically generated planted models. Generally, the proposed algorithm for rank-1 geodesics converges in only a few iterations, while larger k requires an increasing number of iterations to converge. These experiments were initialized randomly, but were still able to recover the true geodesic with error at the level of the additive noise. We occasionally see the algorithm converge to poor local minima and fail to recover the true geodesic. We leave it a future work to make the algorithm more robust to these instances and to provide theoretical bounds on geodesic recovery.

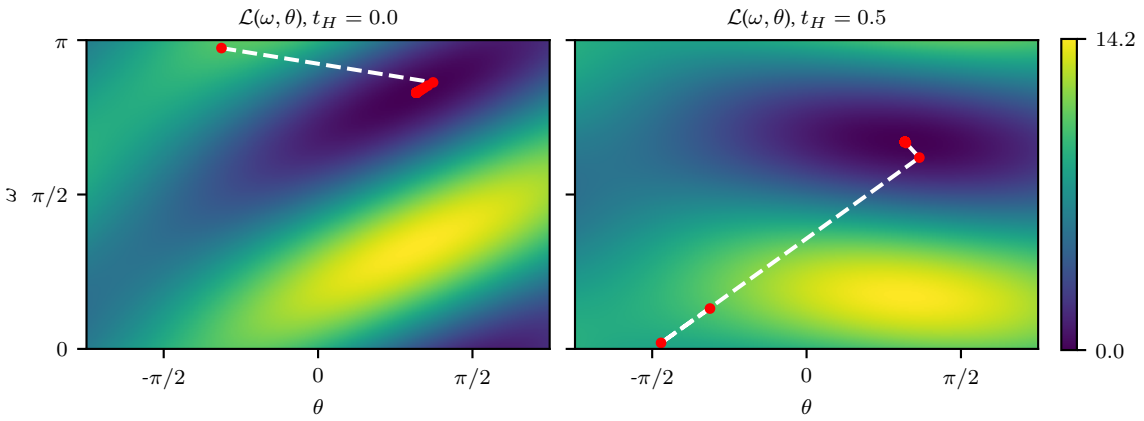


FIG 7.6 – The loss function for learning a rank-1 geodesic in 2D, where we have made the transformation $\boldsymbol{H} = [\cos(\omega); \sin(\omega)]$ and $\boldsymbol{Y} = [-\sin(\omega); \cos(\omega)]$. Note that the y-axes are π-periodic. (Left) The unaltered loss function and associated algorithm iterates (red). The iterates approach the minimum slowly, as the structure of the loss is not well aligned with the coordinate directions. (Right) The loss after making the transformation $\tilde{t}_i = t_i - t_{\boldsymbol{H}}$ for $t_{\boldsymbol{H}} = 0.5$. The iterates fully converge in only a few iterations. Both sets of iterations are initialized at equivalent points.

Rank-1 subspace in $\mathbb{R}^2$ To provide some simple intuition for our problem, we present the algorithm applied to learning a rank-1 subspace in two dimensions. In this special case, $\boldsymbol{\Theta}$ is a scalar θ, and $\boldsymbol{H}$ and $\boldsymbol{Y}$ can be parameterized by a single scalar rotation ω (up to a sign flip in $\boldsymbol{Y}$, which we can absorb into θ). Then $\boldsymbol{H} = [\cos(\omega); \sin(\omega)]$ and $\boldsymbol{Y} = [-\sin(\omega); \cos(\omega)]$. Our loss (7.4) then simplifies to a two-dimensional function of θ and ω

$$\mathcal{L}(\boldsymbol{U}) = -\sum_{i=1}^{T} r_{i,1} \cos(2\theta t_i - \phi_{i,1} + 2\omega) + b_{i,1} + c, \tag{7.20}$$

where $r_{i,1}, \phi_{i,1}, b_{i,1}$ are defined as previously, but with $[\boldsymbol{H}\, \boldsymbol{Y}] = \boldsymbol{I}$ and c is the same constant from (7.4).

FIG 7.6 (left) shows this loss function on some noisy synthetic data with iterates of the proposed algorithm shown in red. Many iterations take small steps as the minimizers of θ and ω are very interdependent.

Intuitively, this behavior may be because the optimal starting subspace $\boldsymbol{H}$ is very dependent on the arc length θ of the geodesic, as the method will naturally want to center the geodesic to minimize error. If we instead parameterized our geodesic by its center subspace, the resulting $\boldsymbol{H}$ minimizer would hopefully be more independent of arc length. This can be done by applying the proposed method after first transforming the time points by letting $\tilde{t}_i = t_i - t_{\boldsymbol{H}}$, where $t_{\boldsymbol{H}} \in [0, 1]$ is the point along the geodesic equal to $\boldsymbol{H}$. **FIG** 7.6

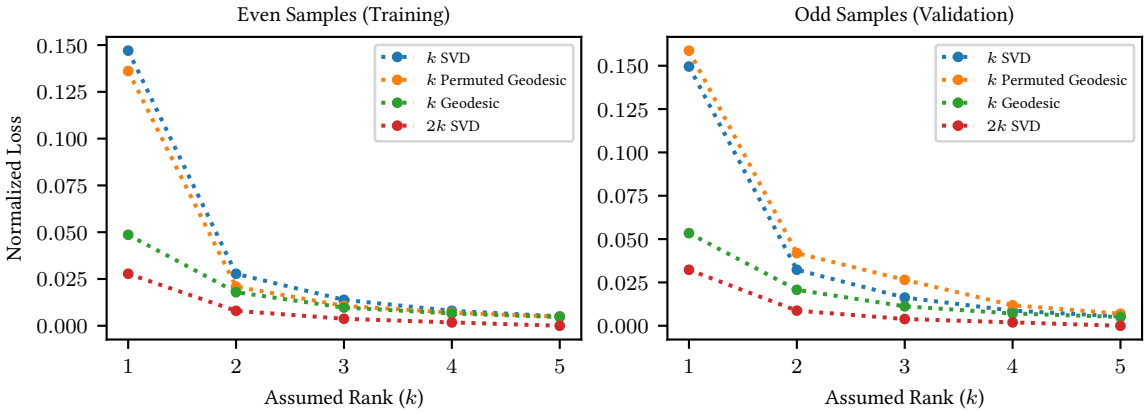


FIG 7.7 – Loss for OSSI dynamic fMRI data. 10 fast time points were collected in a vector ($d = 10$) and modeled across slow time. We fit the various subspace models first investigated in **FIG** 7.4 on just the even samples of slow time and then validated this loss on odd samples of slow time. Note that the loss does not penalize models for overfitting, and so it is expected that the rank-$2k$ SVD is lower even in validation (when $k = 5$ the rank of the SVD equals d and the loss is 0). In the $k = 1$ case, the proposed method has a loss close to rank-$2k$ SVD and permuting the data produces a loss closer to rank-k SVD, which is similar to **FIG** 7.4 (top, left). From this behavior, we infer that the data likely has rank-1 geodesic structure.

(right) shows the loss and associated iterates of setting $t_{\boldsymbol{H}} = 1/2$. The proposed algorithm converges in only a few iterations.

7.3.2 fMRI Data

We show the effectiveness of the proposed method by applying it to (fully anonymized) dynamic functional MRI data collected with IRB approval. An effective data model for fMRI could be applied as part of an advanced image reconstruction algorithm, allowing for reduced scan times and higher temporal resolution without sacrificing image quality. We leave a full investigation of joint reconstruction and modeling as future work and, here, show only the viability of this model on fMRI data. In particular, we apply the proposed method on data collected with an oscillating steady state imaging (OSSI) [34] acquisition on a 3T GE MR750 scanner. Appendix B.2.1 and [34] provide more details on acquisition and reconstruction parameters and example data. The OSSI acquisition rapidly cycles through 10 (t_{fast}) different acquisition settings and then repeats this (t_{slow}) for the duration of the scan. During the scan, subject breathing and scanner drift lead to slowly varying subspace changes that we hypothesized are suitable for a geodesic model. The scanner drift

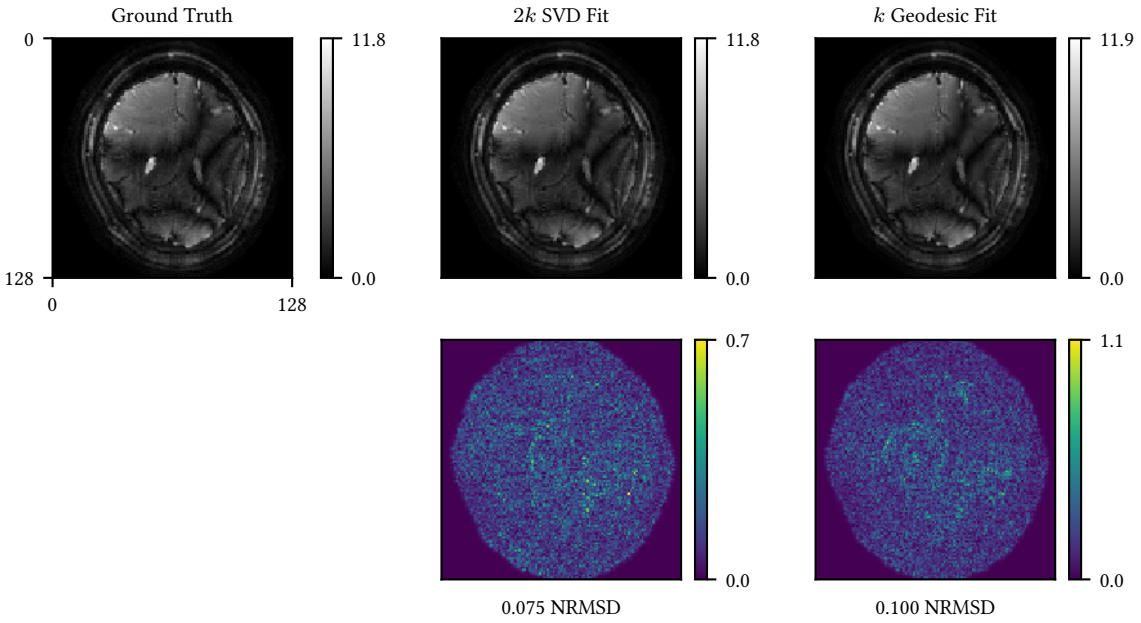
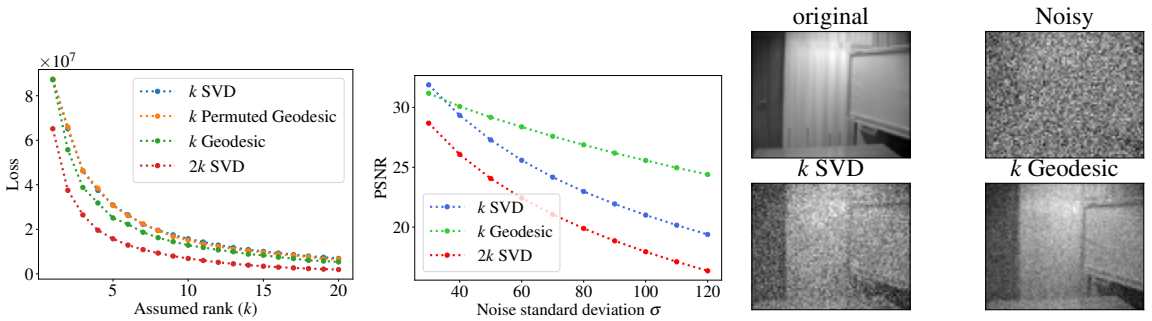


FIG 7.8 – Comparison of OSSI magnitude images reconstructed with a static $2k$ subspace *vs.* the geodesic model for a particular slow and fast time point. For each pixel, fast time points were collected in a vector and a rank-1 geodesic was fit across even slow time points for training. Odd slow time points were then projected onto the geodesic for test. The bottom row shows the magnitude difference maps of the reconstructions against the true test point. The geodesic model appears to have smoothed the image, possibly removing more noise than the SVD. Despite being more temporally constrained, the proposed method has a similar test loss.



(a) (b) (c)

FIG 7.9 – Quantitative evaluation of geodesic subspace model for video data. In (a) loss from (7.4) is plotted for a video sequence containing 260 frames/images. Loss is plotted against different values of assumed rank of data k. In (b) we added AWGN to the video data and then apply rank-k SVD, rank-$2k$ SVD, and the geodesic model to denoise the noisy version of video with $k = 10$ and $\ell = 4$. (c) Visual example of denoising frame 125 in the Curtain video sequence with AWGN of $\sigma = 110$. The geodesic model was able to denoise the noisy image more effectively than SVD.

is approximately linear in time, so equally spaced t_i values seems reasonable. The measurements are dynamic, high dimensional, and show redundant anatomical structure. As is common for image subspace models, we model a spatial patch of data.

FIG 7.7 shows the loss of applying the proposed method for a variety of ranks. Similar to **FIG** 7.4, we show a comparison to rank-k and rank-$2k$ SVD and the proposed method on the data permuted. From this figure, we can see that the OSSI data appears to be well modeled by a rank 1 geodesic; the proposed method with rank 1 performs similarly to rank-k SVD and permuting the data significantly increases the loss to that of rank-k SVD.

FIG 7.8 provides comparison to ground truth and rank-$2k$ SVD for the geodesic OSSI reconstruction presented in **FIG** 7.7 on a single slow and fast time point. The proposed geodesic model fits nearly as well despite being a more constrained model.

7.3.3 Video denoising

In this section we apply the geodesic data model for a video denoising application. The video sequence used in this experiment is the first 260 frames of the Curtain video dataset [52, Section V-A]. The video sequence has variations due to a moving curtain and a person entering the scene at the end of the video. See more details and results for other videos in Appendix B.2.2.

We start by showing that the geodesic model is a good choice for this video data. Along similar lines as the fMRI data, we achieve this goal by computing loss (representation error) for approximating the video using rank-k SVD, rank-$2k$ SVD, the proposed geodesic method, and applying the geodesic method after reordering/permuting the frames in video sequence. Again, the rationale behind permuting the frames is that if there is no temporal correlation in the frames then permuting the data would not have any negative impact on the loss. **FIG** 7.9 shows the training loss as a function of the rank k and the PSNR of the denoised video as a function of added noise level. The training loss for the geodesic model lies in between k and $2k$ cases, similar to the simulated data. More importantly, applying the geodesic model to permuted data degrades the loss, confirming that there is temporal correlation for the geodesic model to exploit. We add additive white Gaussian noise (AWGN) with different values of standard deviation σ to the video sequence and applied rank-k SVD, rank-$2k$ SVD, and the proposed geodesic subspace model to denoise the video sequence. The quality of the denoised image is measured using the peak signal to noise ratio (PSNR), defined as:

$$\text{PSNR} = 20\log_{10}\left(\frac{255}{\frac{1}{\sqrt{d\ell}}\|\boldsymbol{X}_i - \hat{\boldsymbol{X}}_i\|_{\text{F}}}\right).$$

FIG 7.9 shows that rank-$2k$ SVD has the worst denoising performance (due to overfitting the very large amount of noise) while the proposed geodesic method has the best performance in the noisiest regime. For visual evidence, **FIG** 7.9c illustrates the denoising performance of rank-k SVD and geodesic model for frame 125 in the sequence when noise of $\sigma = 110$ is added. Again, similar to the simulated data, the superior denoising performance of geodesic model is quite evident.

7.4 Algorithm Extensions and Future Work

7.4.1 Piecewise Geodesics

While the OSSI data can be modeled well by a rank-1 geodesic, **FIG** 7.10 provides some initial evidence that a piecewise rank-1 geodesic may be a better model. In this figure, we first modeled it with a single rank-1 geodesic (as described in this chapter), and plotted value of the loss function at each data point $\boldsymbol{X}_i$ (on the left). From this we can see that the loss varies in a W shape across the data points, which may indicate that a piecewise geodesic is more appropriate. Then we tried two rank-1 geodesics, fitting the first half

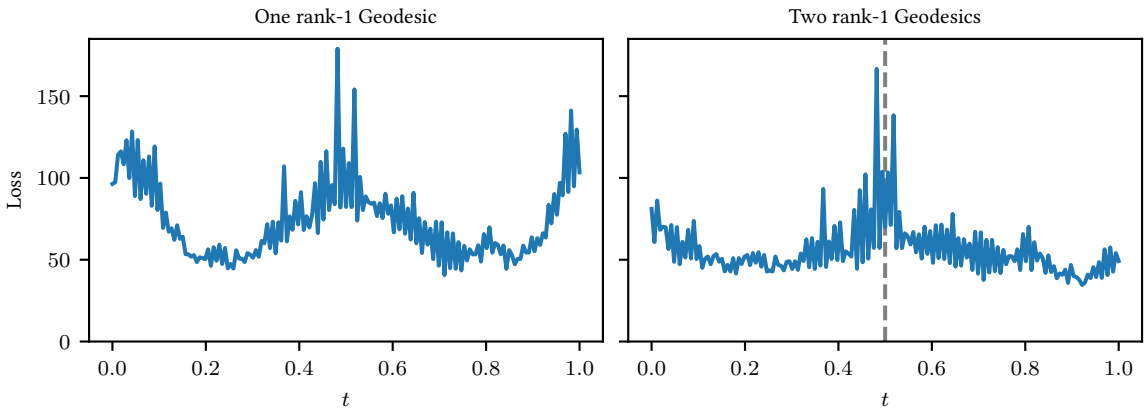


FIG 7.10 – **(Left)** The loss broken down for each time point t for fitting a rank-1 geodesic on OSSI fMRI data. In addition to a seemingly noisy component, the loss also exhibits systematic variations. We hypothesize that these systematic variations result from the limitations of a single geodesic to fit a curve. **(Right)** Instead of fitting a single rank-1 geodesic, we model the data as a piecewise geodesic by fitting a rank-1 geodesic on each half of the data. The resulting loss retains the noisy variations from the single geodesic model, but removes most of the systematic variation.

and second half of the data to separate rank-1 geodesics. When we plot the loss for this model, the shape of the loss is somewhat flatter. A more detailed investigation of methods for identifying knots in a piecewise geodesic model is of great interest for future work.

Even when we know the knots of a piecewise geodesic model, applying the described geodesic estimation algorithm on each segment individually is not guaranteed to produce a continuous piecewise geodesic. If we let $\boldsymbol{U}_j(t)$ represent the jth geodesic segment, then we would like $\text{span}(\boldsymbol{U}_j(1)) = \text{span}(\boldsymbol{U}_{j+1}(0))$, or equivalently $\boldsymbol{U}_j(1)\boldsymbol{U}_j(1)' = \boldsymbol{U}_{j+1}(0)\boldsymbol{U}_{j+1}(0)'$, $\forall j = 1 \dots J-1$. Combining this constraint with our loss yields the following optimization problem for piecewise geodesics

$$\min_{\boldsymbol{U}_j} -\sum_{j=1}^{J}\sum_{i=1}^{T} \left\| \boldsymbol{X}'_{j,i}\boldsymbol{U}_j(t_i) \right\|_{\text{F}}^2 \tag{7.21}$$
$$\text{s.t. } \boldsymbol{U}_j(1)\boldsymbol{U}_j(1)' = \boldsymbol{U}_{j+1}(0)\boldsymbol{U}_{j+1}(0)' \quad \forall j = 1 \dots J-1.$$

We can simplify this problem by relaxing the constraint to a penalty. If we optimize the loss using a BCD approach on each segment, then the penalized loss for the jth segment

is

$$\min_{\boldsymbol{U}_j} -\sum_{i=1}^{T} \|\boldsymbol{X}_i' \boldsymbol{U}_j(t_i)\|_{\mathrm{F}}^2 + \lambda \|\boldsymbol{U}_j(0)\boldsymbol{U}_j(0)' - \boldsymbol{U}_{j-1}(1)\boldsymbol{U}_{j-1}(1)'\|_{\mathrm{F}}^2 + \lambda \|\boldsymbol{U}_j(1)\boldsymbol{U}_j(1)' - \boldsymbol{U}_{j+1}(0)\boldsymbol{U}_{j+1}(0)'\|_{\mathrm{F}}^2 \tag{7.22}$$

$$\min_{\boldsymbol{U}_j} -\sum_{i=1}^{T} \|\boldsymbol{X}_i' \boldsymbol{U}_j(t_i)\|_{\mathrm{F}}^2 - \lambda \|\boldsymbol{U}_{j-1}(1)'\boldsymbol{U}_j(0)\|_{\mathrm{F}}^2 - \lambda \|\boldsymbol{U}_{j+1}(0)'\boldsymbol{U}_j(1)\|_{\mathrm{F}}^2, \tag{7.23}$$

where $\boldsymbol{X}_i$ is the data on just the jth segment. Intuitively, this loss can be minimized using the same updates described previously, but with extra data $\tilde{\boldsymbol{X}}_1 \triangleq \sqrt{\lambda}\boldsymbol{U}_{j-1}(1)$ and $\tilde{\boldsymbol{X}}_T \triangleq \sqrt{\lambda}\boldsymbol{U}_{j+1}(0)$ at each end of the geodesic, $t_1 = 0$ and $t_T = 1$ respectively.

This formulation will perform better on a continuous piecewise geodesic than just applying the proposed method to each geodesic individually, but this approach will only approximate a continuous piecewise geodesic when λ is sufficiently large (though too large of λ will cause the BCD algorithm to disregard the data and stall suboptimally). An effective approach may be to start with $\lambda = 0$ and slowly increase it as the algorithm converges. This may be sufficient since we are only approximating real data as being derived from a piecewise geodesic and still are able to leverage the extra data. In a large data regime, enforcing the constraint too strictly may only lead to larger modeling error. Thus, a penalized formulation allows us to balance the fit to the model and the data. Alternatively, an ADMM [13] approach could be developed to satisfy the constraint strictly in a similar manner.

7.4.2 Inverse Problems

We can apply the geodesic model to solving inverse problems by including the geodesic loss[1] (7.3) as a regularization term

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \min_{\boldsymbol{H},\boldsymbol{Y}\in\mathcal{V},\boldsymbol{\Theta}} \min_{\{\boldsymbol{g}_{i,1}\dots\boldsymbol{g}_{i,\ell}\}_{i=1}^{T}} \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \frac{\beta}{2}\sum_{i=1}^{T}\sum_{j=1}^{\ell} \|\boldsymbol{P}_{i,j}\boldsymbol{x} - \boldsymbol{U}(t_i)\boldsymbol{g}_{i,j}\|_2^2, \tag{7.24}$$

where $\boldsymbol{P}_{i,j}$ extracts the jth patch of the ith time point. Note that letting $[\boldsymbol{P}_{i,1}\boldsymbol{x}\dots\boldsymbol{P}_{i,\ell}\boldsymbol{x}] = \boldsymbol{X}_i$ (and similarly for $\boldsymbol{g}_{i,j}$, $\boldsymbol{G}_i$) would allow us to rewrite this regularization term as we have before. As such, a BCD approach yields the same updates for $\boldsymbol{H}$, $\boldsymbol{Y}$, and $\boldsymbol{\Theta}$ described previously.

[1]We note that the constant in geodesic loss (7.4) depends on the data $\boldsymbol{X}$ and so is not equivalent to (7.3) in this case.

This minimization problem is similar to the UTL cost (2.21) with the non-smooth $\boldsymbol{z}_i$ minimization replaced by a simple quadratic minimization of $\boldsymbol{g}_i$ and $\boldsymbol{U}(t_i) = \boldsymbol{T}'$. We could, thus, apply analogous methods for updating $\boldsymbol{x}$ as given by this closed form solution (*c.f.* (2.26))

$$\boldsymbol{x}^{(n+1)} = \left(\boldsymbol{A}'\boldsymbol{A} + \beta \sum_{i=1}^{T}\sum_{j=1}^{\ell} \boldsymbol{P}_{i,j}'\boldsymbol{P}_{i,j}\right)^{-1} \left(\boldsymbol{A}'\boldsymbol{y} + \beta \sum_{i=1}^{T}\sum_{j=1}^{\ell} \boldsymbol{P}_{i,j}'\boldsymbol{U}(t_i)\boldsymbol{U}(t_i)'\boldsymbol{P}_{i,j}\boldsymbol{x}^{(n)}\right). \tag{7.25}$$

$\boldsymbol{P}_{i,j}'\boldsymbol{P}_{i,j}$ is a diagonal matrix which enables efficient updates, particularly when $\boldsymbol{A}'\boldsymbol{A}$ is easily diagonalizable (*e.g.*, single-coil MRI). Here, we have substituted the optimal $\boldsymbol{g}_{i,j}^{(n)} = \boldsymbol{U}(t_i)'\boldsymbol{P}_{i,j}\boldsymbol{x}^{(n)}$ after deriving the solution.

Another method for updating $\boldsymbol{x}$ would be to first project out the dependence on $\boldsymbol{g}_{i,j}$ before deriving the solution

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \min_{\boldsymbol{H},\boldsymbol{Y}\in\mathcal{V},\boldsymbol{\Theta}} \frac{1}{2}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \frac{\beta}{2}\sum_{i=1}^{T}\sum_{j=1}^{\ell} \|(\boldsymbol{I} - \boldsymbol{U}(t_i)\boldsymbol{U}(t_i)')\,\boldsymbol{P}_{i,j}\boldsymbol{x}\|_2^2. \tag{7.26}$$

This is a Tikhonov regularization problem which penalizes signal energy outside the current geodesic estimate. The solution to the $\boldsymbol{x}$ update can then be given in closed form as

$$\hat{\boldsymbol{x}} = \left(\boldsymbol{A}'\boldsymbol{A} + \beta \sum_{i=1}^{T}\sum_{j=1}^{\ell} \boldsymbol{P}_{i,j}'\,(\boldsymbol{I} - \boldsymbol{U}(t_i)\boldsymbol{U}(t_i)')\,\boldsymbol{P}_{i,j}\right)^{-1} \boldsymbol{A}'\boldsymbol{y}. \tag{7.27}$$

It is unclear if there is an efficient method for computing this update on large-scale problems with efficiently diagonalizable $\boldsymbol{A}'\boldsymbol{A}$ akin to (7.25). We could apply an efficient iterative minimization algorithm for quadratics, such as nonlinear conjugate gradient, using the following gradient

$$\boldsymbol{A}'\,(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}) + \beta \sum_{i=1}^{T}\sum_{j=1}^{\ell} \boldsymbol{P}_{i,j}'\,(\boldsymbol{I} - \boldsymbol{U}(t_i)\boldsymbol{U}(t_i)')\,\boldsymbol{P}_{i,j}\boldsymbol{x}. \tag{7.28}$$

Computing the gradient before substituting the optimal $\boldsymbol{g}_{i,j}$ yields the same expression.

For problems with efficiently diagonalizable $\boldsymbol{A}'\boldsymbol{A}$, it is unclear how iterating (7.25) (corresponding to a two block coordinate quadratic minimization) would compare to nonlinear conjugate gradient on a single quadratic minimization problem with gradient (7.28). We leave it as future work to compare these data updates and to apply this regularizer on inverse problems such as OSSI reconstruction.

7.4.3 Estimating sample times

The proposed method requires knowledge of the sample time points t_i. This is reasonable for temporal data with actual sample times, but there may be problems where we do not know the sample times or we want to model an unknown varying velocity across the geodesic. In these cases, we would need a method to estimate suitable t_i.

We can minimize the loss (7.10) with respect t_i

$$\hat{t}_i = \underset{t_i : t_{i-1} < t_i < t_{i+1}}{\arg\min} -\sum_{j=1}^{k} r_{i,j} \cos(2\theta_j t_i - \phi_{i,j}) + b_{i,j}, \tag{7.29}$$

where we have assumed we at least know the order of the samples. This ordering and the associated bounds constraint makes the problem not fully separable across sample index i. Because this dependence is only with neighboring sample times, we can minimize the sample times with a two-block descent on the even and odd sample time indices. We can also fix, without loss of generality, the first and last sample time to be 0 and 1, respectively. Alternatively, normalization can be done after by appropriately adjusting $\boldsymbol{H}$, $\boldsymbol{Y}$ and $\boldsymbol{\Theta}$.

The minimization can be done by following the same derivation carried out for the $\boldsymbol{\Theta}$ update in Section 7.2.2 by replacing θ_j with t_i, t_i with θ_j, and $\sum_{i=1}^{T}$ with $\sum_{j=1}^{k}$, and projecting each majorize-minimize iteration into bounds

$$t_i^{(n+1)} = \mathcal{P}_{[t_{i-1}, t_{i+1}]} \left(t_i^{(n)} - \frac{\sum_{j=1}^{k} \dot{\tilde{f}}_{i,j}\left(t_i^{(n)}\right)}{\sum_{j=1}^{k} \tilde{w}_{\tilde{f}_{i,j}}\left(t_i^{(n)}\right)} \right). \tag{7.30}$$

7.4.4 Accelerating $(\boldsymbol{H}, \boldsymbol{Y})$ Estimation

Estimating $(\boldsymbol{H}, \boldsymbol{Y})$ each iteration requires computing an SVD which is relatively expensive compared to other update steps. The proposed $(\boldsymbol{H}, \boldsymbol{Y})$ update constructs a linear majorizer on $\mathbb{R}^{d \times 2k}$ even though $\mathcal{V}^{d \times 2k}$ would be sufficient. Thus, there may be room for a tighter majorizer and more efficient update. We leave deriving such a majorizer to future work. Because our current majorizer is valid over a larger set, it may be possible to relax the non-convex Stiefel constraint to $\|[\boldsymbol{H}\ \boldsymbol{Y}]\|_2^2 \leq 1$ and produce the same iterates. Such a convex relaxation seems to hold for the static PCA loss and might similarly hold here to ease analysis.

Another option to speed up the algorithm is to update $\boldsymbol{H}$ and $\boldsymbol{Y}$ under a stricter constraint that supersedes the Stiefel constraint. For example, we could update $\boldsymbol{H}$ and $\boldsymbol{Y}$

by rotating their corresponding columns within their span

$$\boldsymbol{H}^{(n+1)}(\boldsymbol{\Omega}) = \boldsymbol{H}^{(n)}\cos(\boldsymbol{\Omega}) + \boldsymbol{Y}^{(n)}\sin(\boldsymbol{\Omega}) \tag{7.31}$$

$$\boldsymbol{Y}^{(n+1)}(\boldsymbol{\Omega}) = -\boldsymbol{H}^{(n)}\sin(\boldsymbol{\Omega}) + \boldsymbol{Y}^{(n)}\cos(\boldsymbol{\Omega})\,. \tag{7.32}$$

Here, $\boldsymbol{\Omega}$ is a diagonal matrix with diagonal elements ω_j representing rotations for each $\boldsymbol{H}$,$\boldsymbol{Y}$ column pair. Under this parameterization, we can update our loss by minimizing $\boldsymbol{\Omega}$ and $\boldsymbol{\Theta}$. We note that such a update is not sufficient on its own for updating $\boldsymbol{H}$ and $\boldsymbol{Y}$, but can augment the SVD-based update. For this parameterization, the resulting minimization problem and loss function for fixed $\boldsymbol{H}^{(n)}$ and $\boldsymbol{Y}^{(n)}$ is then separable for each jth diagonal element of $\boldsymbol{\Theta}$ and $\boldsymbol{\Omega}$

$$\hat{\theta}_j, \hat{\omega}_j = \underset{\theta_j,\omega_j}{\arg\min} - \sum_{i=1}^{T} r_{i,j}\cos(2\theta_j t_i - \phi_{i,j} + 2\omega_j) + b_{i,j}. \tag{7.33}$$

As one might expect, letting $\omega_j = 0$ yields the same loss (7.10). Associated parameters $r_{i,j}, \phi_{i,j}$ and $b_{i,j}$ are computed with respect to $\boldsymbol{H}^{(n)}$ and $\boldsymbol{Y}^{(n)}$.

This 2D loss function can be minimized simply with a BCD approach. The $\boldsymbol{\theta}$ update is the same as described previously (just with an additional $2\omega_j$ added to each $\phi_{i,j}$). The ω_j update can be given in closed-form as

$$\hat{\omega}_j = \frac{1}{2}\arctan2\left(-\sum_{i=1}^{T} r_{i,j}\sin(2\theta_j t_i - \phi_{i,j}), \sum_{i=1}^{T} r_{i,j}\cos(2\theta_j t_i - \phi_{i,j})\right). \tag{7.34}$$

Conveniently, alternating between these two updates does not require updating the more expensive parameters to compute $r_{i,j}, \phi_{i,j}$.

By augmenting the $\boldsymbol{H}$, $\boldsymbol{Y}$ update with an interleaved $\boldsymbol{\Omega}$, $\boldsymbol{\Theta}$ update, we were able to minimize rank-1 geodesics in a single iteration when tested on synthetic data. Improvements on synthetic problems with $k > 1$ were generally more marginal. How this update affects the radius of initialization under which the algorithm will converge to the global minimizer is left to future work.

Another interesting insight from reparameterizing the loss in 2D this way is that it can help us understand why having $t_{\boldsymbol{H}} = 1/2$ seems to lead to faster convergence than $t_{\boldsymbol{H}} = 0$ (see **FIG** 7.6). We need fewer updates when $\hat{\omega}_j$ is less dependent on θ_j, *i.e.*, when

$\frac{\mathrm{d}\hat{\omega}_j}{\mathrm{d}\theta_j} \approx 0$. Using some small angle approximations

$$\frac{\mathrm{d}\hat{\omega}_j}{\mathrm{d}\theta_j} \approx \frac{-\sum_{i=1}^{T} 2r_{i,j}(t_i - \hat{t}_{\boldsymbol{H}})}{\sum_{i=1}^{T} 2r_{i,j}} = 0 \tag{7.35}$$

$$\hat{t}_{\boldsymbol{H}} = \frac{\sum_{i=1}^{T} r_{i,j} t_i}{\sum_{i=1}^{T} r_{i,j}}. \tag{7.36}$$

So by choosing a $t_{\boldsymbol{H}}$ close to this weighted average, we need fewer alternations between $\boldsymbol{\Omega}$ (or full $\boldsymbol{H}$,$\boldsymbol{Y}$ if we do not include $\boldsymbol{\Omega}$) and $\boldsymbol{\Theta}$. We could update $t_{\boldsymbol{H}}$ every iteration with the current $r_{i,j}$ to minimize the number of inner iterations, but $t_{\boldsymbol{H}} = 1/2$ tends to be very close to optimal when the data is generated symmetrically and only 3 or so inner iterations are required. It is unclear if it would be worth the extra computation since it would require recomputing $\boldsymbol{H}$ and $\boldsymbol{Y}$.

7.5 Conclusion, Discussion, and Future Work

This work proposed a model and algorithm for dynamic subspace estimation, or batch-computed subspace tracking. The proposed method is sample efficient and the optimization requires no hyperparameters beyond the assumed rank of the data. The model and method are applicable to real data, as shown on dynamic fMRI data and video data, and the single geodesic we studied here represents a major building block for a very general piecewise geodesic model.

The model and loss proposed here are non-convex and, thus, the algorithmic results can be sensitive to initialization. In our experiments, we used an SVD for initialization, and this was generally sufficient to recover the true geodesic in our synthetic experiments. **FIG** 7.2 exhibits some brighter patches in the $T > 2k$ regime where a few instances appear to have converged to poor local minima. These instances seem relatively uncommon for the geodesics considered here. We leave it as future work to develop even better initialization and theoretical bounds on geodesic recovery.

In this chapter we were able to show the monotonicity of the updates. Empirically, we also see this algorithms iterates converge to local minimizers. We leave it as future work to prove this convergence to a stationary point. Ravishankar and Bresler [67, Section 4] prove convergence of an BCD algorithm with a Stiefel constraint and may be a good starting point.

While we did not make explicit assumptions on the additive noise, minimizing our loss will yield the maximum likelihood estimate for white Gaussian noise. If the true noise

distribution has heavy tails or if the data has many outliers, a more robust loss will likely perform better. We leave it as future work to investigate fitting the geodesic model with a robust loss, such as a Huber function.

Finally, subspace tracking is often applied to problems where data is modeled as it arrives, *e.g.*, in array processing and communications. It would therefore also be of great interest to develop a streaming algorithm for the piecewise geodesic model.

CHAPTER 8

Conclusion

This thesis has investigated the application of adaptive regularizers to inverse problems in imaging. Chapters 3 and 4 investigated and extended blind UTL for application to light-field imaging and it was shown that this regularizer could outperform a data-less regularizer on these light-field reconstructions. We attempted to further improve the strength of blind UTL with several blind sparsity threshold heuristics, but the results were inconsistent across the dataset. Chapter 5 than compared blind UTL to its own data-driven counterpart, and we showed that the adaptive regularizer was able to perform better on the mixed dataset of brain and knee MRI scans.

In Chapter 6, we considered larger adaptive image models. Namely, we applied neural networks to model patches of an image, similar to UTL. Optimizing these networks on the patches of a single image proved difficult and settling in poor local minima was common.

In Chapter 7, we considered the dynamic subspace estimation problem. We proposed a novel model for dynamic subspaces, namely the piecewise geodesic model, and developed efficient majorize-minimize updates for estimating the geodesic. The proposed model fit well on dynamic fMRI data and video data. This work presented an initial investigation into the viability of this model. Having shown its viability on real data, many exciting avenues for future work include extending the model further, such as with change-point estimation or by applying it to real-world applications. The model can be applied as an reconstruction-time adaptive regularizer, jointly learning a piecewise geodesic while reconstructing a dynamic image. Investigating the effectiveness of this model as an adaptive regularizer is an exciting next step for this work in particular.

This thesis has motivated the use of adaptive regularization when a high-quality and representative training dataset is not available. In practice, drawing this line can be ambiguous. When is my dataset representative enough? When should I reach for an adaptive regularizer over a trained model in practice? Chapter 5 made an initial empirical investigation of this question for unitary transform learning models and found that for such a limited model, blind learning is almost always as good or better at image recovery. At the

time of this work, we had limited access to a brain image dataset (the brain dataset of this chapter was collected by the author). This small dataset necessitated working with less data-intensive models such as the unitary transform model.

Since then, fastMRI [92] has released a brain dataset comparable in size to their knee dataset. With such a large dataset, an interesting avenue of future work would be repeat the experiments of Chapter 5 using larger models such as deep neural networks. While it is very likely that a U-Net trained exclusively on knee images could out perform an adaptive regularizer like UTL or deep image prior at reconstructing knee images, how would it compare if reconstructing an image it was not trained on, such as a brain image? How many brain images does it need to have seen in training to outperform an adaptive approach, and does this differ for deep networks that incorporate the forward model, such as MODL [2]? And what about other forms of heterogeneity? Are T1-weighted images sufficient for training a network used for reconstructing T2-weighted images of the same anatomy? In many ways, MRI seems like the ideal modality to empirically investigate heterogeneous datasets as the different kinds of heterogeneity are often well labeled (*e.g.*, T1, brain, 3T GE scanner) and images within a certain class are relatively homogeneous. A better understanding of how robust data-driven models are to these differences would help better inform the choice of when an adaptive regularizer should be used in practice.

In addition to these questions, much more work can be done to improve each of these individual regularizers. For transform learning, it would be interesting to further investigate other transform constraints outside of unitary matrices. Chapter 5 presented some initial results that even an unconstrained transform will not learn a zero matrix when optimized using BCD. While we may still want some constraint, this results indicates we may have more flexibility in its choice. The other question this result naturally rises is whether BCD is the best approach for minimizing a transform learning problem. The non-convexity of these problems means that our choice of optimization scheme can significantly influence the limit points of the sequence generated by the algorithm. This source of implicit bias was not investigated in this thesis and, as such, it would be interesting to both (1) compare different optimization strategies and (2) better quantify the performance of BCD on UTL using a planted model.

For the geodesic subspace model of Chapter 7, there are many ways the model could be improved. For geodesics with rank greater than 1, the algorithm is slow and a further investigation of ways to accelerate the $\boldsymbol{H}, \boldsymbol{Y}$ update, in particular, is needed. Another important next step with regards to the optimization strategy is proving its convergence to a stationary point. Beyond this, it would also be interesting to develop a theory of global recovery under some set of nontrivial conditions. Finally, Chapter 7 focused on learning a

single geodesic. Expanding this model to true connected piecewise geodesics and applying it to real data is of great interest. Section 7.4 presents an initial investigation of both this and other improvements.

Adaptive image models and regularization methods are valuable tools for reconstructing images from corrupted undersampled measurements. The methods investigated in this thesis require no training dataset and are designed to be computationally tractable for use at the time of reconstruction. As this thesis has shown, designing such methods can be challenging, but they offer a distinct advantages over classical models when a high-quality and representative dataset is not available. For these reasons, there are still many exciting opportunities to improve these models going forward.

APPENDIX A

Derivation of Discrete Focal Stack System Model

In this appendix, we expound on the discretization of the focal stack system model first presented by Nien [60]. To describe a continuous-space light field discretely, we must first decide on a discrete basis for the light field. For brevity, we let $\boldsymbol{x} = (x, y)$ and $\boldsymbol{u} = (u, v)$. A simple choice is to model continuous-space light fields as piece-wise constant with rect basis functions

$$L_F(\boldsymbol{x}, \boldsymbol{u}) \approx \sum_{\boldsymbol{p}} \sum_{\boldsymbol{k}} L_F[\boldsymbol{p}, \boldsymbol{k}] \, \mathrm{rect}_{\Delta_{\mathrm{x}}}(\boldsymbol{x} - \boldsymbol{p}\Delta_{\mathrm{x}}) \, \mathrm{rect}_{\Delta_{\mathrm{u}}}(\boldsymbol{u} - \boldsymbol{k}\Delta_{\mathrm{u}}) \, . \tag{A.1}$$

We denote the distance between the aperture plane and image plane of the light field as F. Given a continuous-space light field with plane separation F, we can compute the continuous-space photograph taken with back focal distance F by integrating out the angular dimension, *i.e.*, each point in the photograph is the summation of rays from all angles hitting that point.

$$I_F(\boldsymbol{x}) = \int_{\boldsymbol{u} \in \mathcal{A}} L_F(\boldsymbol{x}, \boldsymbol{u}) \, \mathrm{d}\boldsymbol{u} \, . \tag{A.2}$$

A digital discrete-space photograph is then captured by integrating the photograph against each (square) pixel

$$I_F[\boldsymbol{m}] = \int I_F(\boldsymbol{x}) \, \mathrm{rect}_{\Delta_{\mathrm{x}}}(\boldsymbol{x} - \boldsymbol{m}\Delta_{\mathrm{x}}) \, \mathrm{d}\boldsymbol{x} \, . \tag{A.3}$$

To compute a digital photograph with arbitrary back focal distance κF we must reparameterize a light field L_F as $L_{\kappa F}$. This can be done using geometric optics, and the assumption that there are no occluders or emitters between the two planes (see **FIG** A.1)

$$L_{\kappa F}(\boldsymbol{x}, \boldsymbol{u}) = L_F(\boldsymbol{x}/\kappa + \boldsymbol{u}(1 - 1/\kappa), \boldsymbol{u}). \tag{A.4}$$

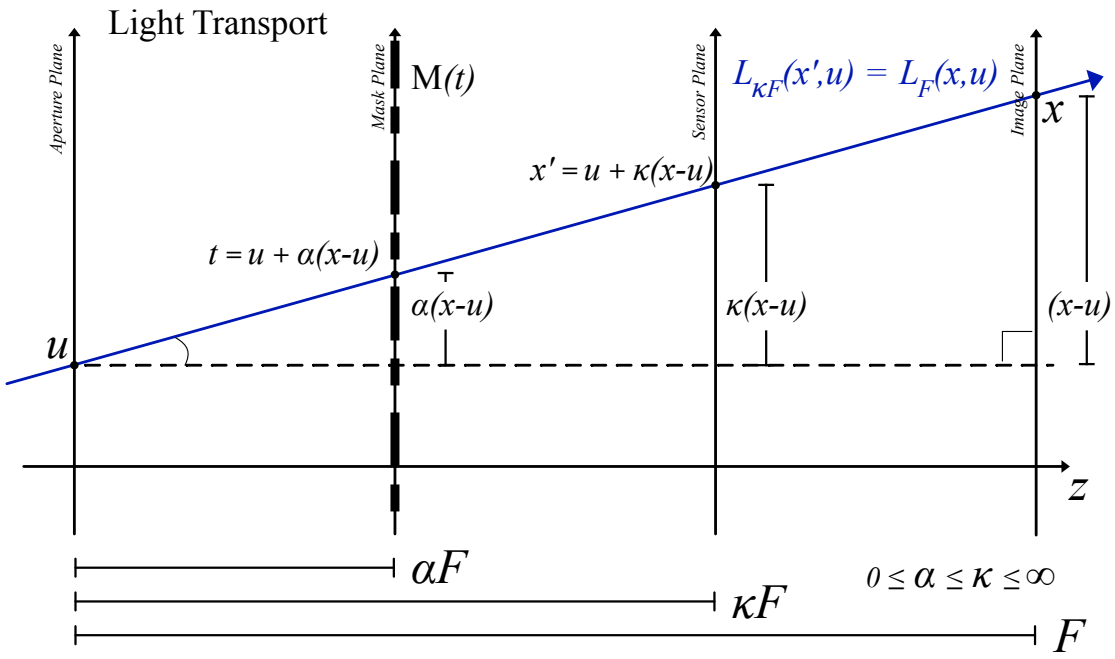


FIG A.1 – A diagram of light transport inside a camera from a geometric optics perspective

We can now relate a captured digital photograph $I_{\kappa F}$ to a latent discrete-space light field L_F. We let $a = 1/\kappa$, $b = (1 - 1/\kappa)$, combine the previous four equations (letting $F = \kappa F$ in equations (A.2),(A.3)), and simplify.

$$
\begin{aligned}
I_{\kappa F}[\boldsymbol{m}] = \int \left(\int \sum_{\boldsymbol{p}} \sum_{\boldsymbol{k}} L_F[\boldsymbol{p}, \boldsymbol{k}] \, \mathrm{rect}_{\Delta_\mathrm{x}}(a\boldsymbol{x} + b\boldsymbol{u} - \boldsymbol{p}\Delta_\mathrm{x}) \, \mathrm{rect}_{\Delta_\mathrm{u}}(\boldsymbol{u} - \boldsymbol{k}\Delta_\mathrm{u}) \, \mathrm{d}\boldsymbol{u} \right) \\
\mathrm{rect}_{\Delta_\mathrm{x}}(\boldsymbol{x} - \boldsymbol{m}\Delta_\mathrm{x}) \, \mathrm{d}\boldsymbol{x} \\
= \sum_{\boldsymbol{p}} \sum_{\boldsymbol{k}} L_F[\boldsymbol{p}, \boldsymbol{k}] \iint \mathrm{rect}_{\Delta_\mathrm{x}}(a\boldsymbol{x} + b\boldsymbol{u} - \boldsymbol{p}\Delta_\mathrm{x}) \, \mathrm{rect}_{\Delta_\mathrm{u}}(\boldsymbol{u} - \boldsymbol{k}\Delta_\mathrm{u}) \, \mathrm{rect}_{\Delta_\mathrm{x}}(\boldsymbol{x} - \boldsymbol{m}\Delta_\mathrm{x}) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{u} \\
= \sum_{\boldsymbol{p}} \sum_{\boldsymbol{k}} L_F[\boldsymbol{p}, \boldsymbol{k}] \iint \mathrm{rect}_{\Delta_\mathrm{x}}(a\boldsymbol{x} + b\boldsymbol{u} - \boldsymbol{p}\Delta_\mathrm{x}) \, \mathrm{rect}_{\Delta_\mathrm{u}}(\boldsymbol{u} - \boldsymbol{k}\Delta_\mathrm{u}) \, \mathrm{rect}_{\Delta_\mathrm{x}}(\boldsymbol{m}\Delta_\mathrm{x} - \boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{u} \\
= \sum_{\boldsymbol{p}} \sum_{\boldsymbol{k}} L_F[\boldsymbol{p}, \boldsymbol{k}] \iint \mathrm{rect}_{\Delta_\mathrm{x}}((a\boldsymbol{x} - \boldsymbol{p}\Delta_\mathrm{x}) + b\boldsymbol{u}) \, \mathrm{rect}_{\Delta_\mathrm{u}}(\boldsymbol{u} - \boldsymbol{k}\Delta_\mathrm{u}) \, \mathrm{rect}_{\Delta_\mathrm{x}}(\boldsymbol{m}\Delta_\mathrm{x} - \boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{u}
\end{aligned}
\tag{A.5}
$$

Perform a change of variables:

$$
\boldsymbol{x}' = a\boldsymbol{x} - \boldsymbol{p}\Delta_\mathrm{x} \quad \mathrm{d}\boldsymbol{x}' = a \, \mathrm{d}\boldsymbol{x} \qquad \boldsymbol{u}' = -\boldsymbol{u} \quad \mathrm{d}\boldsymbol{u}' = -\mathrm{d}\boldsymbol{u} \tag{A.6}
$$

$$
\begin{aligned}
&= \sum_{\boldsymbol{p}} \sum_{\boldsymbol{k}} L_F[\boldsymbol{p}, \boldsymbol{k}] \iint \frac{1}{a} \operatorname{rect}_{\Delta_{\mathrm{x}}}(\boldsymbol{x}' - b\boldsymbol{u}') \operatorname{rect}_{\Delta_{\mathrm{u}}}(-\boldsymbol{k}\Delta_{\mathrm{u}} - \boldsymbol{u}') \\
&\qquad\qquad \operatorname{rect}_{\Delta_{\mathrm{x}}}\left(\boldsymbol{m}\Delta_{\mathrm{x}} - \frac{1}{a}(\boldsymbol{p}\Delta_{\mathrm{x}} + \boldsymbol{x}')\right) \mathrm{d}\boldsymbol{x}' \, \mathrm{d}\boldsymbol{u}' \\
&= \sum_{\boldsymbol{p}} \sum_{\boldsymbol{k}} L_F[\boldsymbol{p}, \boldsymbol{k}] \iint \frac{1}{a} \operatorname{rect}_{\Delta_{\mathrm{x}}}(\boldsymbol{x}' - b\boldsymbol{u}') \operatorname{rect}_{\Delta_{\mathrm{u}}}(-\boldsymbol{k}\Delta_{\mathrm{u}} - \boldsymbol{u}') \\
&\qquad\qquad \operatorname{rect}_{a\Delta_{\mathrm{x}}}(a\boldsymbol{m}\Delta_{\mathrm{x}} - \boldsymbol{p}\Delta_{\mathrm{x}} - \boldsymbol{x}') \, \mathrm{d}\boldsymbol{x}' \, \mathrm{d}\boldsymbol{u} \\
&= \sum_{\boldsymbol{p}} \sum_{\boldsymbol{k}} L_F[\boldsymbol{p}, \boldsymbol{k}] \left\{ \frac{1}{a} \operatorname{rect}_{\Delta_{\mathrm{x}}}(\boldsymbol{x} - b\boldsymbol{u}) * \operatorname{rect}_{\Delta_{\mathrm{u}}}(\boldsymbol{u}) \operatorname{rect}_{a\Delta_{\mathrm{x}}}(\boldsymbol{x}) \right\} \Big|_{\boldsymbol{x}=(a\boldsymbol{m}-\boldsymbol{p})\Delta_{\mathrm{x}}, \boldsymbol{u}=-\boldsymbol{k}\Delta_{\mathrm{u}}} \\
&= \sum_{\boldsymbol{p}} \sum_{\boldsymbol{k}} L_F[\boldsymbol{p}, \boldsymbol{k}] \, g[a\boldsymbol{m} - \boldsymbol{p}, 0 - \boldsymbol{k}] \\
&= \{L_F * g\}[\boldsymbol{m}', \boldsymbol{k}'] \Big|_{\boldsymbol{m}'=a\boldsymbol{m}, \boldsymbol{k}'=0}
\end{aligned}
\tag{A.7}
$$

Thus, a 2D captured photograph can be computed from a 4D discrete-space light field by discretely convolving with kernel

$$
g[\boldsymbol{p}, \boldsymbol{k}] = \left\{ \frac{1}{a} \operatorname{rect}_{\Delta_{\mathrm{x}}}(\boldsymbol{x} - b\boldsymbol{u}) * \operatorname{rect}_{\Delta_{\mathrm{u}}}(\boldsymbol{u}) \operatorname{rect}_{a\Delta_{\mathrm{x}}}(\boldsymbol{x}) \right\} \Big|_{\boldsymbol{x}=\boldsymbol{p}\Delta_{\mathrm{x}}, \boldsymbol{u}=\boldsymbol{k}\Delta_{\mathrm{u}}} \tag{A.8}
$$

and slicing. In practice, The 4D convolution can be performed as a set of 2D convolutions along the slice [60]. The constant a conveys the magnification that occurs at different sensor positions. This is occasionally disregarded ($a \approx 1$) when magnification is minimal (or, equivalently, when a focal stack is collected by changing the focal length of the lens instead of moving the sensor).

I leave it as future work to expound on other system models as necessary. Currently, diffraction is not considered as we have derived the system model from a purely geometric optics perspective. Color is also not directly considered, but can be modeled easily enough by modeling a separate light field for each channel and masking each pixel value in I by its corresponding color.

APPENDIX B

Supplementary Material for "Dynamic Subspace Estimation with Piecewise Geodesics"

B.1 Additional Algorithmic Derivations and Details

B.1.1 Derivation of $(\boldsymbol{H}, \boldsymbol{Y})$ Update as Linear Majorize Minimize Step

Let $\boldsymbol{Q} \triangleq [\boldsymbol{H}\ \boldsymbol{Y}]$ and $\boldsymbol{Z}_i \triangleq [\cos(\boldsymbol{\Theta} t_i); \sin(\boldsymbol{\Theta} t_i)]$. Then our model can be written as

$$\boldsymbol{U}_i = \boldsymbol{H} \cos(\boldsymbol{\Theta} t_i) + \boldsymbol{Y} \sin(\boldsymbol{\Theta} t_i) = \boldsymbol{Q} \boldsymbol{Z}_i. \tag{B.1}$$

To form a linear majorizer for loss with respect to $\boldsymbol{Q}$, we first derive its unconstrained gradient

$$\mathcal{L}(\boldsymbol{Q}) = -\sum_{i=1}^{T} \left\| \boldsymbol{X}_i' \boldsymbol{Q} \boldsymbol{Z}_i \right\|_{\mathrm{F}}^2 + c = -\sum_{i=1}^{T} \mathsf{trace}\{\boldsymbol{Q}' \boldsymbol{X}_i \boldsymbol{X}_i' \boldsymbol{Q} \boldsymbol{Z}_i \boldsymbol{Z}_i'\} + c \tag{B.2}$$

$$\nabla_{\boldsymbol{Q}} \mathcal{L}\big(\bar{\boldsymbol{Q}}\big) = -\sum_{i=1}^{T} 2 \boldsymbol{X}_i \boldsymbol{X}_i' \bar{\boldsymbol{Q}} \boldsymbol{Z}_i \boldsymbol{Z}_i'. \tag{B.3}$$

We can form a linear majorizer for the loss

$$g(\boldsymbol{Q}; \bar{\boldsymbol{Q}}) \triangleq \mathsf{trace}\big\{\boldsymbol{Q}' \nabla_{\boldsymbol{Q}} \mathcal{L}\big(\bar{\boldsymbol{Q}}\big)\big\} + c \tag{B.4}$$

$$\geq \mathcal{L}\big(\bar{\boldsymbol{Q}}\big)\,. \tag{B.5}$$

Note that $g(\bar{\boldsymbol{Q}};\bar{\boldsymbol{Q}}) = \mathcal{L}\big(\bar{\boldsymbol{Q}}\big)$, and it is linear and continuous. The above inequality only needs to hold for $\boldsymbol{Q} \in \mathcal{V}^{d\times 2k}$. It currently holds for $\mathbb{R}^{d\times 2k}$ so there is room for a tighter majorizer.

Following the work of Breloy *et al.* [14], we can write $g(\boldsymbol{Q};\bar{\boldsymbol{Q}}) = -2\mathsf{trace}\big\{\boldsymbol{Q}' R(\bar{\boldsymbol{Q}})\big\} + c$ for matrix function $R(\bar{\boldsymbol{Q}}) = \sum_{i=1}^{T} \boldsymbol{X}_i\boldsymbol{X}_i'\bar{\boldsymbol{Q}}\boldsymbol{Z}_i\boldsymbol{Z}_i'$. We can then minimize the linear majorizer g with a Stiefel manifold constraint simply by projecting its negative gradient $R(\bar{\boldsymbol{Q}})$ onto the Stiefel manifold. The update is then given by

$$\hat{\boldsymbol{Q}} = \underset{\boldsymbol{Q}\in\mathcal{V}^{d\times 2k}}{\arg\min} \left\| R(\bar{\boldsymbol{Q}}) - \boldsymbol{Q} \right\|_{\mathrm{F}}^2 \tag{B.6}$$

$$= \underset{\boldsymbol{Q}\in\mathcal{V}^{d\times 2k}}{\arg\min} \left\| \left(\sum_{i=1}^{T} \boldsymbol{X}_i\boldsymbol{X}_i'\bar{\boldsymbol{Q}}\boldsymbol{Z}_i\boldsymbol{Z}_i' \right) - \boldsymbol{Q} \right\|_{\mathrm{F}}^2 \tag{B.7}$$

$$= \boldsymbol{W}\boldsymbol{V}', \tag{B.8}$$

$$\text{where } \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{V}' = \sum_{i=1}^{T} \boldsymbol{X}_i\boldsymbol{X}_i'\bar{\boldsymbol{Q}}\boldsymbol{Z}_i\boldsymbol{Z}_i'$$

$$= \sum_{i=1}^{T} \left[\boldsymbol{X}_i\hat{\boldsymbol{G}}_i{}' \cos(\boldsymbol{\Theta}t_i) \quad \boldsymbol{X}_i\hat{\boldsymbol{G}}_i{}' \sin(\boldsymbol{\Theta}t_i) \right],$$

where in the last line we have let $\hat{\boldsymbol{G}}_i{}' = \boldsymbol{X}_i'\boldsymbol{U}_i^{(n)} = \boldsymbol{X}_i'\bar{\boldsymbol{Q}}\boldsymbol{Z}_i$.

We can derive this same update as a block coordinate update on $\boldsymbol{Q}$ and $\{\boldsymbol{G}_i\}_{i=1}^T$. We start with our loss (7.4) without projecting out $\{\boldsymbol{G}_i\}_{i=1}^T$ and substitute our geodesic model (7.2). The $(\boldsymbol{H},\boldsymbol{Y})$ update with fixed $\boldsymbol{\Theta}$ and $\{\boldsymbol{G}_i\}_{i=1}^T$ can be minimized by recognizing it as a generalized Procrustes problem

$$\hat{\boldsymbol{H}},\hat{\boldsymbol{Y}} = \underset{\boldsymbol{H},\boldsymbol{Y}\in\mathcal{V}^{d\times k},\, \boldsymbol{H}'\boldsymbol{Y}=\boldsymbol{0}}{\arg\min} \sum_{i=1}^{T} \left\| \boldsymbol{X}_i - \left(\boldsymbol{H}\cos(\boldsymbol{\Theta}t_i) + \boldsymbol{Y}\sin(\boldsymbol{\Theta}t_i)\right)\boldsymbol{G}_i \right\|_{\mathrm{F}}^2 \tag{B.9}$$

$$[\hat{\boldsymbol{H}}\ \hat{\boldsymbol{Y}}] = \underset{[\boldsymbol{H}\ \boldsymbol{Y}]\in\mathcal{V}^{d\times 2k}}{\arg\min} \sum_{i=1}^{T} \left\| \boldsymbol{X}_i - [\boldsymbol{H}\ \boldsymbol{Y}] \begin{bmatrix} \cos(\boldsymbol{\Theta}t_i) \\ \sin(\boldsymbol{\Theta}t_i) \end{bmatrix} \boldsymbol{G}_i \right\|_{\mathrm{F}}^2 \tag{B.10}$$

$$= \boldsymbol{W}\boldsymbol{V}', \tag{B.11}$$

$$\text{where } \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{V}' = \sum_{i=1}^{T} \boldsymbol{X}_i \left(\begin{bmatrix} \cos(\boldsymbol{\Theta}t_i) \\ \sin(\boldsymbol{\Theta}t_i) \end{bmatrix} \boldsymbol{G}_i \right)'$$

$$= \sum_{i=1}^{T} \left[\boldsymbol{X}_i\boldsymbol{G}_i' \cos(\boldsymbol{\Theta}t_i) \quad \boldsymbol{X}_i\boldsymbol{G}_i' \sin(\boldsymbol{\Theta}t_i) \right].$$

This Procrustes step involves a single SVD of the $d \times 2k$ matrix shown in the last line. While derived on different losses, Updates (B.8) and (B.11) yield the same update for $(\boldsymbol{H}, \boldsymbol{Y})$.

B.1.2 Derivation of $\boldsymbol{\Theta}$ Update

We derive a majorize minimize iteration for $\boldsymbol{\Theta}$. First we simplify the loss and highlight the separability of the loss with respect to the diagonal elements of $\boldsymbol{\Theta}$. We then construct majorizers for each term in the simplified loss using a translated Huber majorizer. The update is then given by minimizing the sum of these majorizers.

We note that while we may refer to $\boldsymbol{\Theta}$ as arc distances, we do not constrain the elements of $\boldsymbol{\Theta}$ to be non-negative. Conceptually, the negative values of $\boldsymbol{\Theta}$ represent walking in the opposite direction on the surface of the Grassmannian, *i.e.*, they are *signed* arc distances.

B.1.2.1 Simplifying the Loss

We start by simplifying the loss

$$\hat{\boldsymbol{\Theta}} = \arg\min_{\boldsymbol{\Theta}} \min_{\{\boldsymbol{G}_i\}} \sum_{i=1}^{T} \left\| \boldsymbol{X}_i - \left(\boldsymbol{H}\cos(\boldsymbol{\Theta} t_i) + \boldsymbol{Y}\sin(\boldsymbol{\Theta} t_i)\right) \boldsymbol{G}_i \right\|_{\mathrm{F}}^2 \tag{B.12}$$

$$= \arg\min_{\boldsymbol{\Theta}} - \sum_{i=1}^{T} \left\| \boldsymbol{X}_i' \left(\boldsymbol{H}\cos(\boldsymbol{\Theta} t_i) + \boldsymbol{Y}\sin(\boldsymbol{\Theta} t_i)\right) \right\|_{\mathrm{F}}^2 \tag{B.13}$$

$$\begin{aligned} = \arg\min_{\boldsymbol{\Theta}} - \sum_{i=1}^{T} \mathsf{trace}\Big\{ \boldsymbol{X}_i' \Big(& \boldsymbol{H}\cos^2(\boldsymbol{\Theta} t_i)\, \boldsymbol{H}' \\ & + 2\,\mathrm{real}\{\boldsymbol{H}\cos(\boldsymbol{\Theta} t_i)\sin(\boldsymbol{\Theta} t_i)\, \boldsymbol{Y}'\} \\ & + \boldsymbol{Y}\sin^2(\boldsymbol{\Theta} t_i)\, \boldsymbol{Y}' \Big) \boldsymbol{X}_i \Big\} \end{aligned} \tag{B.14}$$

$$\begin{aligned} = \arg\min_{\boldsymbol{\Theta}} - \sum_{i=1}^{T} \mathsf{trace}\Big\{ & \cos^2(\boldsymbol{\Theta} t_i)\, \boldsymbol{H}'\boldsymbol{X}_i\boldsymbol{X}_i'\boldsymbol{H} \\ & + 2\cos(\boldsymbol{\Theta} t_i)\sin(\boldsymbol{\Theta} t_i)\,\mathrm{real}\{\boldsymbol{Y}'\boldsymbol{X}_i\boldsymbol{X}_i\boldsymbol{H}\} \\ & + \sin^2(\boldsymbol{\Theta} t_i)\, \boldsymbol{Y}'\boldsymbol{X}_i\boldsymbol{X}_i'\boldsymbol{Y} \Big\} \end{aligned} \tag{B.15}$$

$$\begin{aligned} = \arg\min_{\boldsymbol{\Theta}} - \sum_{i=1}^{T} \sum_{j=1}^{k} & \cos^2(\theta_j t_i)\, [\boldsymbol{H}'\boldsymbol{X}_i\boldsymbol{X}_i'\boldsymbol{H}]_{j,j} \\ & + 2\cos(\theta_j t_i)\sin(\theta_j t_i)\, [\mathrm{real}\{\boldsymbol{Y}'\boldsymbol{X}_i\boldsymbol{X}_i\boldsymbol{H}\}]_{j,j} \\ & + \sin^2(\theta_j t_i)\, [\boldsymbol{Y}'\boldsymbol{X}_i\boldsymbol{X}_i'\boldsymbol{Y}]_{j,j}. \end{aligned} \tag{B.16}$$

We solve the problem of optimizing $\boldsymbol{\Theta}$ by updating each of its diagonal elements θ_j separately. We define the following constants

$$\alpha_{i,j} = [\boldsymbol{H}'\boldsymbol{X}_i\boldsymbol{X}_i'\boldsymbol{H}]_{j,j} \tag{B.17}$$

$$\beta_{i,j} = \text{real}\Big\{[\boldsymbol{Y}'\boldsymbol{X}_i\boldsymbol{X}_i'\boldsymbol{H}]_{j,j}\Big\} \tag{B.18}$$

$$\gamma_{i,j} = [\boldsymbol{Y}'\boldsymbol{X}_i\boldsymbol{X}_i'\boldsymbol{Y}]_{j,j}. \tag{B.19}$$

Our optimization problem for each $j = 1, \ldots, k$ is now

$$\hat{\theta}_j = \arg\min_{\theta_j} -\sum_{i=1}^{T} \alpha_{i,j}\cos^2(\theta_j t_i) + 2\beta_{i,j}\cos(\theta_j t_i)\sin(\theta_j t_i) + \gamma_{i,j}\sin^2(\theta_j t_i)\,. \tag{B.20}$$

Using the following trigonometric identities

$$2\cos(x)\sin(x) = \sin(2x) \tag{B.21}$$

$$\cos^2(x) + \sin^2(x) = 1 \tag{B.22}$$

$$\cos^2(x) = \frac{1}{2}\left(\cos(2x) + 1\right) \tag{B.23}$$

$$a\cos(x) + b\sin(x) = \sqrt{a^2 + b^2}\cos(x - \text{arctan2}(b, a)), \tag{B.24}$$

we further simplify the loss:

$$\hat{\theta}_j = \arg\min_{\theta_j} -\sum_{i=1}^{T} \alpha_{i,j}\cos^2(\theta_j t_i) + \beta_{i,j}\sin(2\theta_j t_i) + \gamma_{i,j}\sin^2(\theta_j t_i) \tag{B.25}$$

$$= \arg\min_{\theta_j} -\sum_{i=1}^{T} (\alpha_{i,j} - \gamma_{i,j})\cos^2(\theta_j t_i) + \beta_{i,j}\sin(2\theta_j t_i) + \gamma_{i,j}\left(\cos^2(\theta_j t_i) + \sin^2(\theta_j t_i)\right) \tag{B.26}$$

$$= \arg\min_{\theta_j} -\sum_{i=1}^{T} (\alpha_{i,j} - \gamma_{i,j})\frac{1}{2}\left(\cos(2\theta_j t_i) + 1\right) + \beta_{i,j}\sin(2\theta_j t_i) + \gamma_{i,j} \tag{B.27}$$

$$= \arg\min_{\theta_j} -\sum_{i=1}^{T} r_{i,j}\cos(2\theta_j t_i - \phi_{i,j}) + b_{i,j}, \tag{B.28}$$

where

$$r_{i,j} = \sqrt{\left(\frac{\alpha_{i,j} - \gamma_{i,j}}{2}\right)^2 + \beta_{i,j}^2} \tag{B.29}$$

$$\phi_{i,j} = \operatorname{arctan2}\left(\beta_{i,j}, \frac{\alpha_{i,j} - \gamma_{i,j}}{2}\right) \tag{B.30}$$

$$b_{i,j} = \frac{\alpha_{i,j} + \gamma_{i,j}}{2}. \tag{B.31}$$

B.1.2.2 Constructing a majorizer

To majorize our loss function, we construct a quadratic majorizer for each term of the form

$$f_{i,j}(\theta_j) \triangleq -r_{i,j}\cos(2\theta_j t_i - \phi_{i,j}) + b_{i,j}. \tag{B.32}$$

To be a majorizer at a point $\bar{\theta}_j$, we require $q_{i,j}\big(\bar{\theta}_j; \bar{\theta}_j\big) = f_{i,j}\big(\bar{\theta}_j\big)$ (equal at the point of construction) and $q_{i,j}\big(\theta_j; \bar{\theta}_j\big) \geq f_{i,j}(\theta_j)$ (greater than or equal to the loss everywhere). This can be achieved with a quadratic of the form

$$q_{i,j}\big(\theta_j; \bar{\theta}_j\big) = f_{i,j}\big(\bar{\theta}_j\big) + \dot{f}_{i,j}\big(\bar{\theta}_j\big)(\theta_j - \bar{\theta}_j) + \frac{1}{2} w_{f_{i,j}}\big(\bar{\theta}_j\big)\,(\theta_j - \bar{\theta}_j)^2, \tag{B.33}$$

where $\dot{f}_{i,j}(\theta_j) = 2r_{i,j}t_i \sin(2\theta_j t_i - \phi_{i,j})$ is the derivative of $f_{i,j}$ and $w_{f_{i,j}}$ is an appropriate curvature (or "weighting") function. A simple option is $w_{f_{i,j}}\big(\bar{\theta}_j\big) = L_{\dot{f}_{i,j}}$ the Lipschitz constant of the derivative. Minimizing the resulting majorizer yields the standard fixed step size gradient descent algorithm. A tighter majorizer will touch our original function at two or more points. Note that $\frac{\phi_{i,j}}{2t_i}$ is a minimizer and our function is symmetric and quasi-convex on the interval $\left[\frac{\phi_{i,j}-\pi}{2t_i}, \frac{\phi_{i,j}+\pi}{2t_i}\right]$ about this point. Our approach will be to construct a curvature function $\bar{w}_{f_{i,j}}$ for points in this interval and periodically extend it to construct the final curvature function $w_{f_{i,j}}$. Because $f_{i,j}$ is symmetric about $\frac{\phi_{i,j}}{2t_i}$, our majorizer will touch at two points when the axis (and minimizer) of $q_{i,j}$ is $\frac{\phi_{i,j}}{2t_i}$, equivalently when its gradient at this point equals zero

$$\dot{q}_{i,j}\left(\frac{\phi_{i,j}}{2t_i}; \bar{\theta}_j\right) = 0. \tag{B.34}$$

Solving for $\bar{w}_{f_{i,j}}\big(\bar{\theta}_j\big)$ yields

$$\bar{w}_{f_{i,j}}\big(\bar{\theta}_j\big) = \frac{\dot{f}_{i,j}\big(\bar{\theta}_j\big)}{\bar{\theta}_j - \frac{\phi_{i,j}}{2t_i}}, \tag{B.35}$$

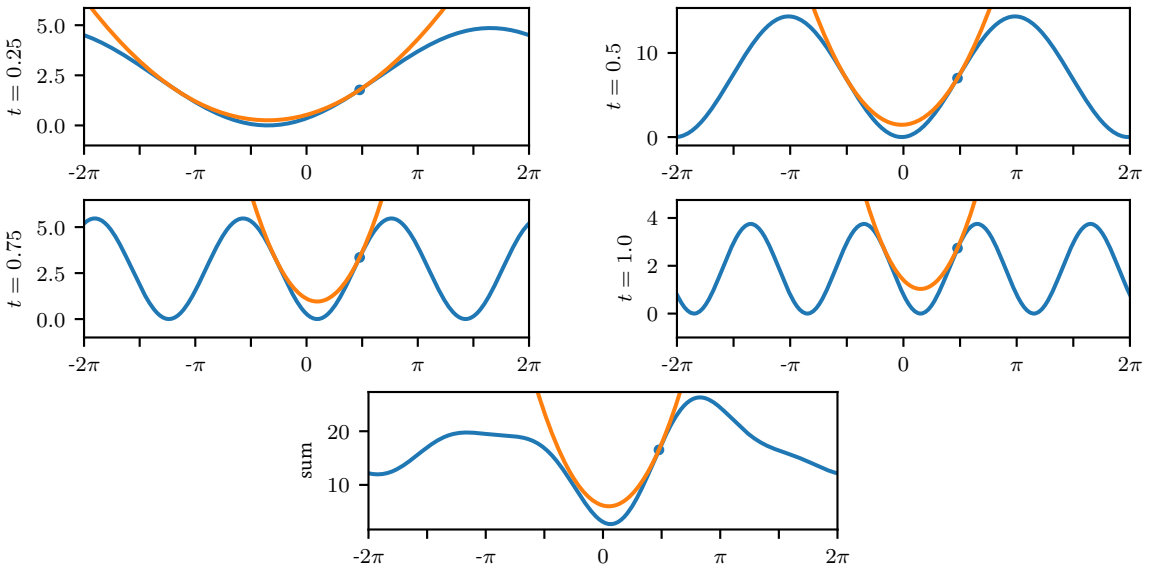


FIG B.1 – An example of four cosines (top two rows, blue) that sum to form the (non-convex) loss for a single θ_j (bottom row, blue). For each cosine function, we construct a quadratic majorizer (top two rows, orange) at a point $\bar{\theta}_j$ (blue dot). The sum of these individual quadratic majorizers form a quadratic majorizer for the loss (bottom, orange) that has a closed-form minimizer. Although the loss is non-convex, distance-minimizing geodesics will have $\theta_j \in [-\pi/2, \pi/2]$. On this interval, the loss is often well-behaved (here, quasi-convex).

which can be recognized as a (translated) Huber curvature function. For the case when $\bar{\theta}_j = \frac{\phi_{i,j}}{2t_i}$, we define $w_{f_{i,j}}\left(\frac{\phi_{i,j}}{2t_i}\right) = 4t_i^2 r_{i,j}$, which is its limit point.

Forming $w_{f_{i,j}}$ by periodically extending $\bar{w}_{f_{i,j}}$ only requires periodically extending the denominator, since the numerator is already periodic. The resulting periodic version of the curvature function is

$$w_{f_{i,j}}\left(\bar{\theta}_j\right) = \begin{cases} \dfrac{\dot{f}_{i,j}\left(\bar{\theta}_j\right)}{\operatorname{mod}\left(\left(\bar{\theta}_j - \frac{\phi_{i,j}}{2t_i}\right) + \frac{\pi}{2t_i}, \frac{2\pi}{2t_i}\right) - \frac{\pi}{2t_i}} & \bar{\theta}_j \neq \frac{\phi_{i,j}+2\pi m}{2t_i}, m \in \mathbb{Z} \\ 4t_i^2 r_{i,j} & \bar{\theta}_j = \frac{\phi_{i,j}+2\pi m}{2t_i}, m \in \mathbb{Z}. \end{cases} \tag{B.36}$$

The final majorizer for our loss function of θ_j is then

$$q_j\left(\theta_j; \bar{\theta}_j\right) = \sum_{i=1}^{T} q_{i,j}\left(\theta_j; \bar{\theta}_j\right). \tag{B.37}$$

FIG B.1 shows an example loss and the constructed majorizer.

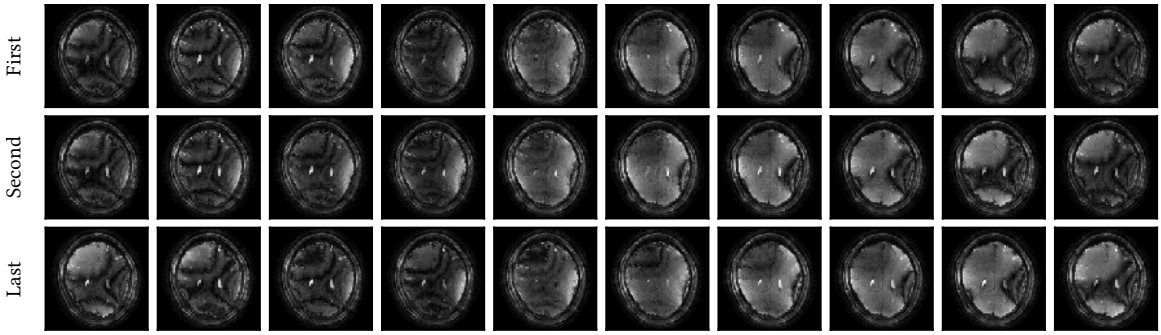


FIG B.2 – A sample of OSSI acquisitions. Each column, referred to as fast time, represents a certain set of acquisition parameters. Each row, referred to as slow time, is a complete cycle through these acquisitions, which is done many times. Here we only show the first, second, and last slow time set. We see that there is little difference between neighboring slow time points, but that over the course of the scan they change more significantly.

B.1.2.3 Minimizing the Majorizer

Because the majorizer is a sum of one-dimensional quadratics, we minimize it by setting its derivative to zero and solving for θ_j

$$\dot{q}_j\left(\theta_j;\bar{\theta}_j\right) = \sum_{i=1}^{T} \dot{q}_{i,j}\left(\theta_j;\bar{\theta}_j\right) \tag{B.38}$$

$$0 = \sum_{i=1}^{T} \dot{f}_{i,j}\left(\bar{\theta}_j\right) + w_{f_{i,j}}\left(\bar{\theta}_j\right)\left(\theta_j - \bar{\theta}_j\right) \tag{B.39}$$

$$= \left(\sum_{i=1}^{T} \dot{f}_{i,j}\left(\bar{\theta}_j\right)\right) + \left(\sum_{i=1}^{T} w_{f_{i,j}}\left(\bar{\theta}_j\right)\right)\left(\theta_j - \bar{\theta}_j\right) \tag{B.40}$$

$$\theta_j = \bar{\theta}_j - \frac{\sum_{i=1}^{T} \dot{f}_{i,j}\left(\bar{\theta}_j\right)}{\sum_{i=1}^{T} w_{f_{i,j}}\left(\bar{\theta}_j\right)}. \tag{B.41}$$

Iteratively constructing majorizers and minimizing them yields the following descent scheme

$$\theta_j^{(n+1)} = \theta_j^{(n)} - \frac{\sum_{i=1}^{T} \dot{f}_{i,j}\left(\theta_j^{(n)}\right)}{\sum_{i=1}^{T} w_{f_{i,j}}\left(\theta_j^{(n)}\right)}. \tag{B.42}$$

B.2 Additional Experiments and Details

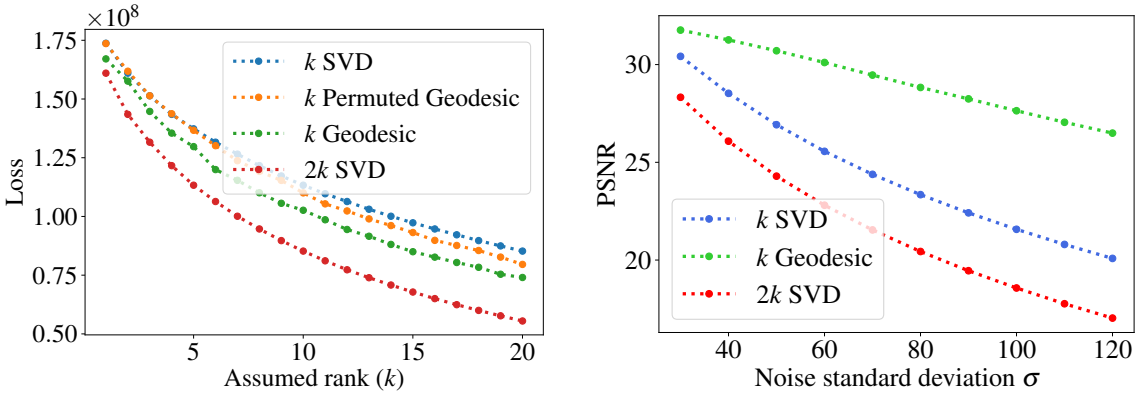
B.2.1 OSSI Dynamic fMRI Dataset Details

The OSSI dynamic fMRI dataset was acquired on a 3T GE MR750 scanner with a 32-channel head coil and is comprised of 167 slow time, 10 fast time and 128 $\times$ 128 spatial samples. The complex data was fully sampled with a variable-density spiral trajectory with $n_i = 8$ interleaves, a densely sampled core, and spiral-out readouts. Detailed OSSI acquisition parameters (TR, n_c, TE, and flip angle) can be found in Guo and Noll [34]. The volunteer was given a left versus right reversing-checkerboard visual stimulus (20 s Left / 20 s Right $\times$ 5 cycles) for 200 s in total. For reconstruction, the k-space data was compressed to 16 virtual coils and ESPIRiT [46] SENSE maps were generated using the BART toolbox [81]. Finally, the images were reconstructed using conjugate gradient SENSE with a Huber potential via the MIRT toolbox [26]. **FIG** B.2 shows the magnitude of a sample of the reconstructed images.

B.2.2 Video Denoising Additional Experiments and Details

The link to the video data from [52, Section V-A] has been broken for a while, but the videos can still be found using the internet archive. We downloaded the videos from [75].

We provide further denoising results for video data in this section. We perform experiments on waterfall video data (in addition to the curtain video we provided in the main text), and the results are presented in **FIG** B.3 and **FIG** B.4b. Same as before, we start by showing that the geodesic model is a good choice for this video data. **FIG** B.3a shows the training loss as a function of the rank k. The training loss for the geodesic model lies in between k and $2k$ cases, as expected. It also has smaller error as compared to permuted data, which is strong evidence that the geodesic model is a reasonable model to consider for the waterfall video sequence. Next, we study the denoising capabilities of rank-k SVD, rank-$2k$ SVD, and rank-k geodesic model by adding AWGN with different values of standard deviation σ to the video sequence and applying these three approaches to remove noise. The quality of the denoised image is measured using the peak signal to noise ratio (PSNR). **FIG** B.3b shows the PSNR of the denoised video as a function of added noise level. Finally, we provide visual evidence of denoising in **FIG** B.4. In **FIG** B.4a frame 125 is shown for denoising the video corrupted with AWGN of $\sigma = 110$. Each image shown is the reconstruction of that frame using each model for denoising, and the PSNR is given at the top of each image. Notice that both PSNR and the perceptual quality of image denoised by the rank-k geodesic model is better than the other two methods, which is a



(a) (b)

FIG B.3 – Quantitative evaluation of geodesic subspace model for waterfall video sequence. In (a) loss from (7.4) is plotted for a video sequence containing 260 frames/images. Loss is plotted against different values of assumed rank of data k. In (b) we added AWGN to the video data and then applied rank-k SVD, rank-$2k$ SVD, and the geodesic model to denoise the noisy version of video with $k = 10$ and $\ell = 4$.

similar trend we observed in **FIG** 7.9b for the Curtain dataset. Next, in **FIG** B.4b we show results from a similar experiment but the noise added is AWGN with $\sigma = 30$, which is significantly lower than the previous experiment. From this experiment we can conclude that the three methods have almost similar performance in low noise settings.

Finally, we display frame 125 for curtain and waterfall sequences for a lower noise (higher SNR) regime in **FIG** B.5. These results show that in lower noise settings rank-$2k$ SVD is able to learn more structure in data and hence has better denoising performance than rank-k SVD and rank-k geodesic model. In contrast, in higher noise settings the structure in the images described by smaller singular values is overwhelmed by noise and rank-$2k$ SVD ends up learning a lot of noise, which diminishes its denoising capabilities.

B.3 Empirical Analysis of Additional Geodesic Estimation Algorithm

In Chapter 7, we parameterized a geodesic by a point on the geodesic $\boldsymbol{H}$, a tangent matrix $\boldsymbol{Y}$, and a set of arc lengths $\boldsymbol{\Theta}$. If we consider only shortest-path geodesics ($\|\boldsymbol{\Theta}\|_2 < \frac{\pi}{2}$), we can describe a geodesic by its start and endpoint, which we will denote $\boldsymbol{A}$ and $\boldsymbol{B}$,

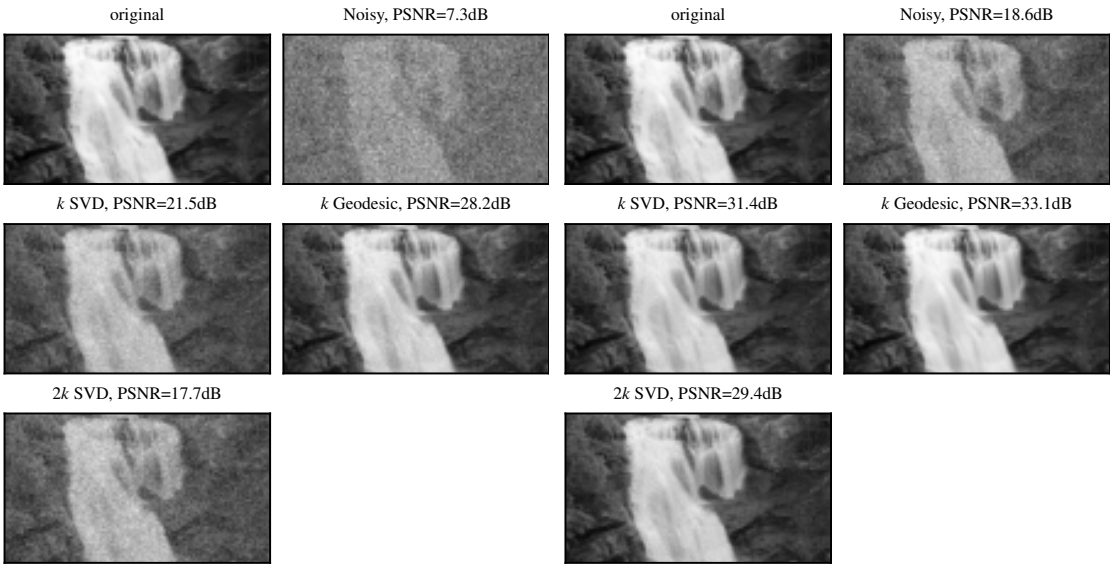


FIG B.4 – Visual example of denoising frame 125 in the waterfall video sequence with AWGN of $\sigma = 110$ in (a) and $\sigma = 30$ in (b). The geodesic model was able to denoise the noisy image more effectively than rank-k SVD and rank-$2k$ SVD in high noise regime in (a). On the other hand for lower noise regime in (b) the denoising performance is quite similar.

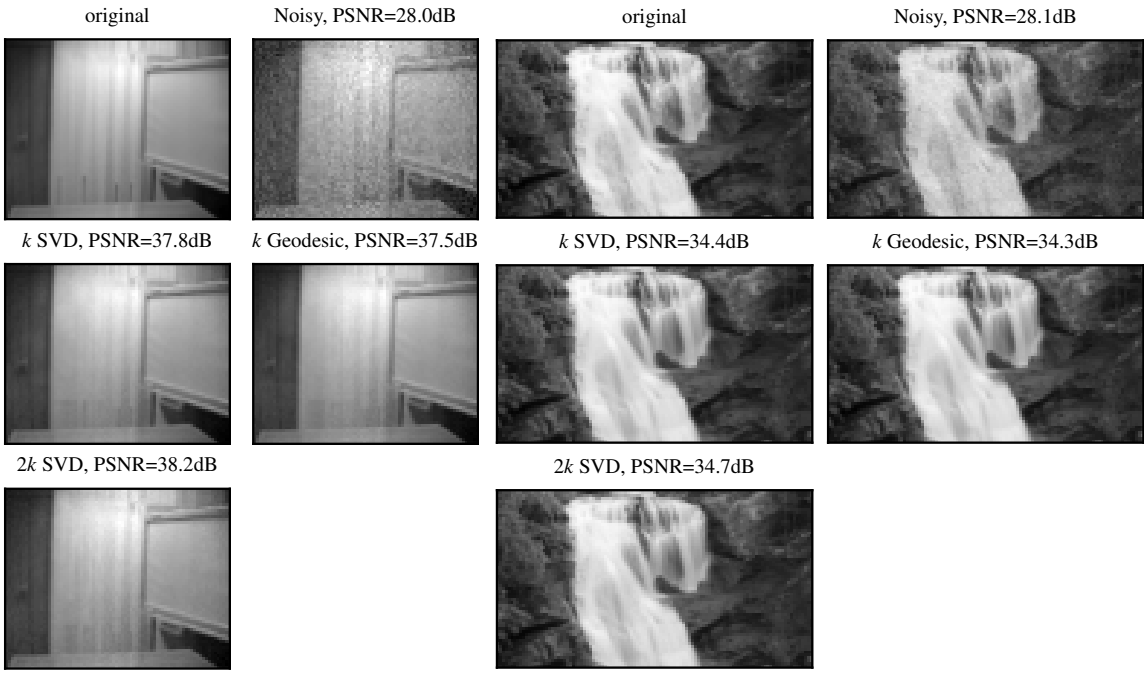


FIG B.5 – When the SNR is higher, the geodesic model and PCA model are more comparable. This figure shows the same frames from both the curtain and waterfall videos and their reconstructions with $\sigma = 10$.

respectively. Our subspaces are then given by

$$\boldsymbol{U}_i = \boldsymbol{U}(t_i; \boldsymbol{A}, \boldsymbol{B}) = \boldsymbol{A}\boldsymbol{Z}\cos(\boldsymbol{\Theta} t_i) + \boldsymbol{Y}\sin(\boldsymbol{\Theta} t_i) \tag{B.43}$$

$$\text{where } \boldsymbol{Y}\tan(\boldsymbol{\Theta})\,\boldsymbol{Z}' \text{ is the SVD of } (\boldsymbol{I} - \boldsymbol{A}\boldsymbol{A}')\,\boldsymbol{B}(\boldsymbol{A}'\boldsymbol{B})^{-1}. \tag{B.44}$$

We will attempt to minimize the geodesic loss (7.3),

$$\hat{\boldsymbol{A}}, \hat{\boldsymbol{B}} = \underset{\boldsymbol{A},\boldsymbol{B}\in\mathcal{G}(k,d)}{\arg\min} \min_{\{\boldsymbol{G}_i\}_{i=1}^T} \sum_{i=1}^T \left\| \boldsymbol{X}_i - \boldsymbol{U}(t_i; \boldsymbol{A}, \boldsymbol{B})\boldsymbol{G}_i \right\|_{\mathrm{F}}^2, \tag{B.45}$$

by applying an alternating update on $\boldsymbol{A}$ and $\boldsymbol{B}$. If we can derive an update on $\boldsymbol{A}$, then we can exploit the symmetry of this geodesic model to also have an update for $\boldsymbol{B}$ since $\boldsymbol{U}(t_i; \boldsymbol{A}, \boldsymbol{B}) = \boldsymbol{U}(1 - t_i; \boldsymbol{B}, \boldsymbol{A})$. Considering just the $\boldsymbol{A}$ update first, we start by substituting the new model (B.43) to the geodesic loss (B.45)

$$\hat{\boldsymbol{A}} = \underset{\boldsymbol{A}\in\mathcal{G}(k,d)}{\arg\min} \sum_{i=1}^T \left\| \boldsymbol{X}_i - \left(\boldsymbol{A}\boldsymbol{Z}\cos(\boldsymbol{\Theta} t_i) + \boldsymbol{Y}\sin(\boldsymbol{\Theta} t_i)\right)\boldsymbol{G}_i \right\|_{\mathrm{F}}^2. \tag{B.46}$$

While the loss has an explicit dependence on $\boldsymbol{A}$, parameters $\boldsymbol{Y}$, $\boldsymbol{\Theta}$, and $\boldsymbol{Z}$ are also implicit complicated functions of $\boldsymbol{A}$. It is unclear what step could be taken minimize this loss with respect to $\boldsymbol{A}$. If we ignore the implicit dependencies on $\boldsymbol{A}$ (ad hoc approximating them as constant), then the above problem is a generalized Procrustes problem

$$\hat{\boldsymbol{A}} = \underset{\boldsymbol{A}\in\mathcal{G}(k,d)}{\arg\min} \sum_{i=1}^T \left\| \left(\boldsymbol{X}_i - \boldsymbol{Y}\sin(\boldsymbol{\Theta} t_i)\,\boldsymbol{G}_i\right) - \boldsymbol{A}\boldsymbol{Z}\cos(\boldsymbol{\Theta} t_i)\,\boldsymbol{G}_i \right\|_{\mathrm{F}}^2 \tag{B.47}$$

$$= \boldsymbol{W}\boldsymbol{V}', \text{ where} \tag{B.48}$$

$$\boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{V}' \text{ is the SVD of } \sum_{i=1}^T \left(\boldsymbol{X}_i - \boldsymbol{Y}\sin(\boldsymbol{\Theta} t_i)\,\boldsymbol{G}_i\right)\left(\boldsymbol{Z}\cos(\boldsymbol{\Theta} t_i)\,\boldsymbol{G}_i\right)'. \tag{B.49}$$

A similar $\boldsymbol{B}$ update can be formulated by swapping the roles of $\boldsymbol{A}$ and $\boldsymbol{B}$ and replacing t_i with $1 - t_i$. While it is unclear if this approximating update is guaranteed to monotonically not increase the loss, empirically it seems to. Indeed, this geodesic estimation algorithm alternating on $\boldsymbol{A}$ and $\boldsymbol{B}$ generally converges in at least an order of magnitude fewer iterations than the algorithm proposed in Chapter 7 on all but the $k = 1$ case (see **FIG** B.6).

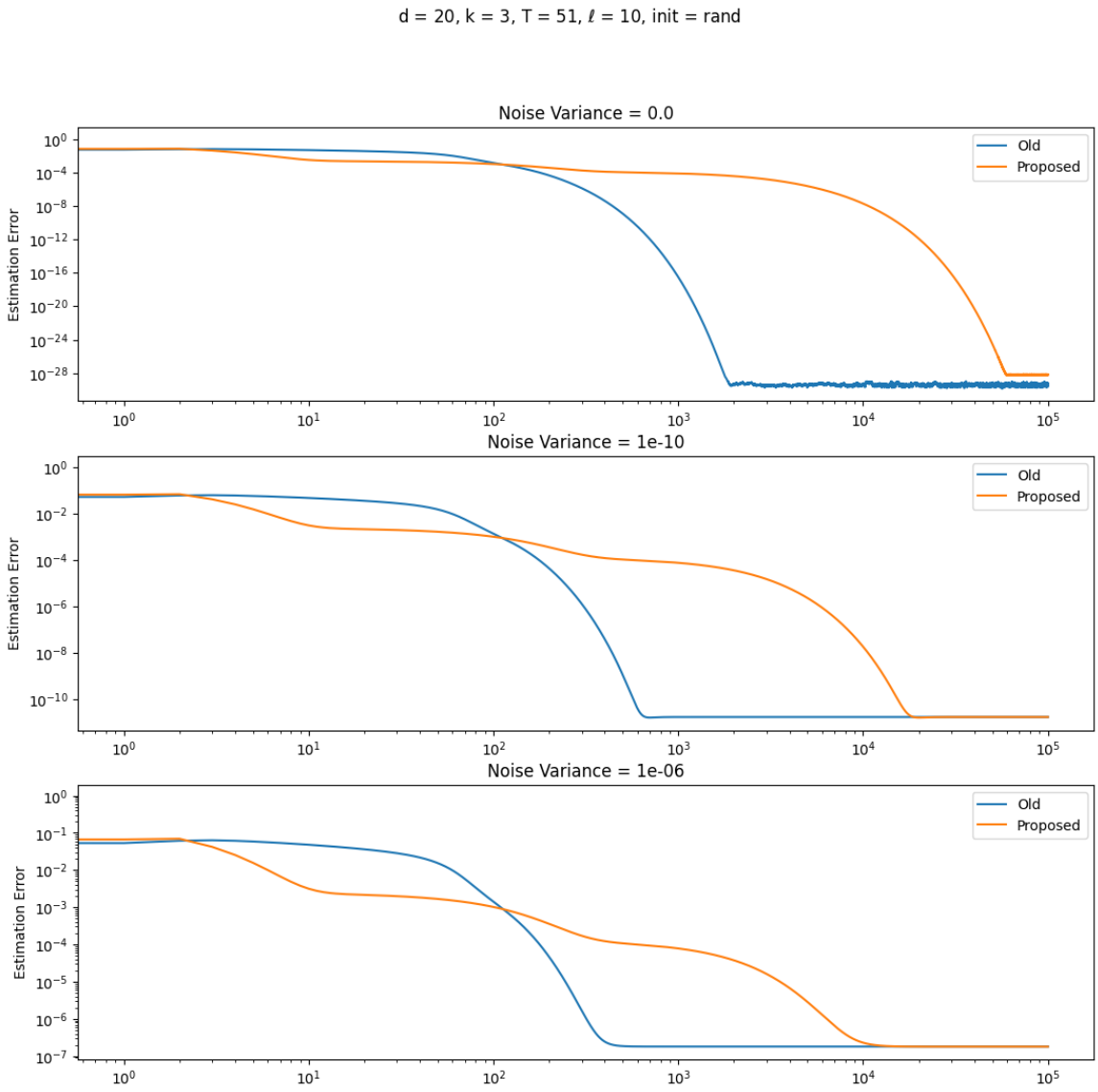


FIG B.6 – Comparison of convergence of geodesic estimation error for two parameterizations of a geodesic. Here "Old" refers to the geodesic model parameterized by its start and endpoint $\boldsymbol{A}$ and $\boldsymbol{B}$. "Proposed" is the geodesic model and associated updates presented in Chapter 7.

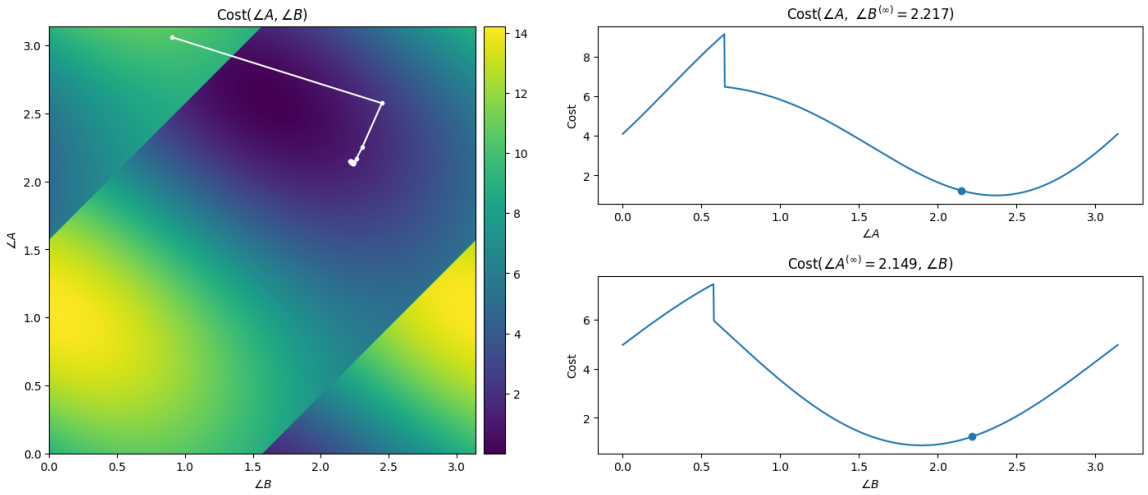


FIG B.7 – A plot of the loss function parameterized by the geodesic endpoints $\boldsymbol{A}, \boldsymbol{B}$, where we project out the dependence on $\{\boldsymbol{G}_i\}$. Because we are in 2D, we can express $\boldsymbol{A}$ and $\boldsymbol{B}$ by their angle with the positive x-axis. On the left, we show the loss surface and the iterates. The discontinuities occur when $|\angle \boldsymbol{A} - \angle \boldsymbol{B}| = \pi/2$. Note that the edges of the plotted loss wraps around in both the vertical and horizontal directions. On the right, we show where $\boldsymbol{A}$ and $\boldsymbol{B}$ got stuck despite not reaching a minimizer of their respective costs.

Unfortunately, there are instances where this algorithm fails to converge to a stationary point of the loss. **FIG** B.7 presents a visual example of a rank-1 geodesic in 2D (the same data shown in **FIG** 7.6) where this algorithm failed to converge to a local minimizer.

BIBLIOGRAPHY

[1] P.-A. Absil, R. Mahony, and R. Sepulchre. "Riemannian geometry of Grassmann manifolds with a view on algorithmic computation". In: *Acta Applicandae Mathematica* 80.2 (2004), pp. 199–220 (cit. on pp. 61, 68).

[2] H. K. Aggarwal, M. P. Mani, and M. Jacob. "MoDL: model-based deep learning architecture for inverse problems". In: *IEEE Trans. Med. Imag.* 38.2 (Feb. 2019), 394–405. DOI: 10.1109/tmi.2018.2865356 (cit. on p. 85).

[3] M. Alain and A. Smolic. "Light Field Denoising by Sparse 5D Transform Domain Collaborative Filtering". In: *Proceedings of the IEEE Workshop on Multimedia Signal Processing (MMSP)*. Oct. 2017, pp. 1–6. DOI: 10.1109/MMSP.2017.8122232 (cit. on pp. 23, 24).

[4] M. Alain and A. Smolic. "Light Field Super-Resolution via LFBM5D Sparse Coding". In: *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. Oct. 2018, pp. 2501–2505. DOI: 10.1109/ICIP.2018.8451162 (cit. on p. 23).

[5] Z. Allen-Zhu and Y. Li. "Follow the compressed leader: faster online learning of eigenvectors and faster MMWU". In: *Proc. 34th Int. Conf. Mach. Learning*. JMLR. org. 2017, pp. 116–125 (cit. on p. 62).

[6] C. D. Bahadir, A. Q. Wang, A. V. Dalca, and M. R. Sabuncu. *Deep-Learning-Based Optimization of the Under-Sampling Pattern in MRI*. July 2019. arXiv: 1907.11374. Pre-published (cit. on p. 42).

[7] L. Balzano, Y. Chi, and Y. Lu. "Streaming PCA and Subspace Tracking: The Missing Data Case". In: *Proc. of IEEE Special Issue on Rethinking PCA for Modern Datasets: Theory, Algorithms, and Applications* (2018) (cit. on p. 62).

[8] J. Bergstra, D. Yamins, and D. D. Cox. *Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms*. 2013. URL: http://hyperopt.github.io/hyperopt/ (visited on 07/23/2019) (cit. on pp. 26, 45).

[9] C. J. Blocker, I. Y. Chun, and J. A. Fessler. "Low-Rank Plus Sparse Tensor Models for Light-Field Reconstruction From Focal Stack Data". In: *Proceedings of the IEEE*

Workshop on Image, Video, and Multidimensional Signal Processing (IVMSP). 2018, pp. 1–5. DOI: 10.1109/IVMSPW.2018.8448509 (cit. on p. 23).

[10] C. J. Blocker and J. A. Fessler. "Blind Unitary Transform Learning for Inverse Problems in Light-Field Imaging". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. Oct. 2019, pp. 3933–3942. DOI: 10.1109/ICCVW.2019.00487 (cit. on pp. 3, 22).

[11] C. J. Blocker, H. Raja, J. A. Fessler, and L. Balzano. *Dynamic Subspace Estimation with Piecewise Geodesics*. submitted, in review. 2022 (cit. on p. 3).

[12] A. Bora, A. Jalal, E. Price, and A. G. Dimakis. "Compressed Sensing Using Generative Models". In: *Proceedings of the International Conference on Machine Learning (ICML)*. Ed. by D. Precup and Y. W. Teh. Vol. 70. PMLR, Aug. 2017, pp. 537–546. DOI: 10.5555/3305381.3305437 (cit. on pp. 51, 55, 56).

[13] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Found. & Trends in Machine Learning* 3.1 (2010), 1–122. DOI: 10.1561/2200000016 (cit. on pp. 7, 78).

[14] A. Breloy, S. Kumar, Y. Sun, and D. P. Palomar. "Majorization-minimization on the Stiefel manifold with application to robust sparse PCA". In: *IEEE Trans. Sig. Proc.* 69 (2021), 1507–20. DOI: 10.1109/TSP.2021.3058442 (cit. on pp. 64, 66, 91).

[15] J. H. R. Chang, C. Li, B. Póczos, B. V. K. Vijaya Kumar, and A. C. Sankaranarayanan. "One Network to Solve Them All — Solving Linear Inverse Problems Using Deep Projection Models". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2017, pp. 5889–5898. DOI: 10.1109/ICCV.2017.627 (cit. on p. 51).

[16] Y. Chi, Y. C. Eldar, and R. Calderbank. "Petrels: Parallel subspace estimation and tracking by recursive least squares from partial observations". In: *IEEE Trans. Signal Process.* 61.23 (2013), pp. 5947–5959 (cit. on p. 62).

[17] I. Y. Chun and J. A. Fessler. "Convolutional dictionary learning: acceleration and convergence". In: *IEEE Trans. Im. Proc.* 27.4 (Apr. 2018), 1697–712. DOI: 10.1109/TIP.2017.2761545 (cit. on p. 55).

[18] D. Colton and R. Kress. *Inverse acoustic and electromagnetic scattering theory*. Vol. 93. Berlin: Springer-Verlag, 1992, pp. x + 305. ISBN: 3-540-55518-8 (cit. on p. 38).

[19] P. Comon and G. H. Golub. "Tracking a few extreme singular values and vectors in signal processing". In: *Proceedings of the IEEE* 78.8 (1990), pp. 1327–1343 (cit. on p. 62).

[20] C. Crockett and J. A. Fessler. "Bilevel methods for image reconstruction". In: *Found. & Trends in Sig. Pro.* (2021). Submitted (cit. on p. 48).

[21] D. Dansereau and L. T. Bruton. "A 4-D Dual-Fan Filter Bank for Depth Filtering in Light Fields". In: *IEEE Transactions on Signal Processing* 55.2 (Feb. 2007), pp. 542–549. ISSN: 1053-587X. DOI: 10.1109/TSP.2006.885733 (cit. on p. 23).

[22] D. G. Dansereau, D. L. Bongiorno, O. Pizarro, and S. B. Williams. "Light Field Image Denoising Using a Linear 4D Frequency-Hyperfan All-In-Focus Filter". In: *Proceedings of SPIE Computational Imaging XI*. Vol. 8657. Feb. 2013. DOI: 10.1117/12.2002239 (cit. on p. 23).

[23] E. Dib, M. Le Pendu, X. Jiang, and C. Guillemot. "Super-Ray Based Low Rank Approximation for Light Field Compression". In: *Proceedings of the IEEE Data Compression Conference*. Mar. 2019, pp. 369–378. ISBN: 978-1-7281-0657-1. DOI: 10.1109/DCC.2019.00045 (cit. on p. 23).

[24] A. Edelman, T. A. Arias, and S. T. Smith. "The geometry of algorithms with orthogonality constraints". In: *SIAM J. Matrix. Anal. Appl.* 20.2 (1998), 303–53. DOI: 10.1137/S0895479895290954 (cit. on p. 61).

[25] R. A. Farrugia, C. Galea, and C. Guillemot. "Super Resolution of Light Field Images Using Linear Subspace Projection of Patch-Volumes". In: *IEEE Journal of Selected Topics in Signal Processing* 11.7 (Oct. 2017), pp. 1058–1071. ISSN: 1932-4553. DOI: 10.1109/JSTSP.2017.2747127 (cit. on p. 23).

[26] J. A. Fessler. *Michigan image reconstruction toolbox (MIRT) for Matlab*. Available from http://web.eecs.umich.edu/~fessler. 2016 (cit. on p. 97).

[27] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. *DeepStereo: Learning to Predict New Views From the World's Imagery*. June 2015. arXiv: 1506.06825. Pre-published (cit. on p. 23).

[28] D. R. Fuhrmann. "A geometric approach to subspace tracking". In: *Asilomar Conf. Signals, Syst. and Comput.* Vol. 1. IEEE. 1997, pp. 783–787 (cit. on pp. 59, 62, 63).

[29] A. K. Funai, J. A. Fessler, D. T. B. Yeo, V. T. Olafsson, and D. C. Noll. "Regularized field map estimation in MRI". In: *IEEE Trans. Med. Imag.* 27.10 (Oct. 2008), 1484–94. DOI: 10.1109/TMI.2008.923956 (cit. on p. 65).

[30] D. Gilton, G. Ongie, and R. Willett. "Learned Patch-Based Regularization for Inverse Problems in Imaging". In: *Proceedings of the IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. Dec. 2019, pp. 211–215. DOI: 10.1109/CAMSAP45676.2019.9022624 (cit. on p. 52).

[31] G. Golub and V. Pereyra. "Separable nonlinear least squares: the variable projection method and its applications". In: *Inverse Prob.* 19.2 (Apr. 2003), R1–26. DOI: 10.1088/0266-5611/19/2/201 (cit. on p. 63).

[32] N. M. Gottschling, V. Antun, B. Adcock, and A. C. Hansen. *The Troublesome Kernel: Why Deep Learning for Inverse Problems Is Typically Unstable*. 2020. arXiv: 2001.01258 [cs.LG]. Pre-published (cit. on p. 8).

[33] H. Gu, B. Yaman, K. Ugurbil, S. Moeller, and M. Akçakaya. "Compressed Sensing MRI with ℓ_1-Wavelet Reconstruction Revisited Using Modern Data Science Tools". In: *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. 2021, pp. 3596–3600. DOI: 10.1109/EMBC46164.2021.9630985 (cit. on p. 40).

[34] S. Guo and D. C. Noll. "Oscillating steady-state imaging (OSSI): A novel method for functional MRI". In: *Mag. Res. Med.* 84.2 (Aug. 2020), 698–712. DOI: 10.1002/mrm.28156 (cit. on pp. 73, 97).

[35] S. Haghighatshoar and G. Caire. "Low-complexity massive MIMO subspace estimation and tracking from low-dimensional projections". In: *IEEE Trans. Signal Process.* 66.7 (2018), pp. 1832–1844 (cit. on pp. 59, 62).

[36] K. Hammernik, T. Klatzer, E. Kobler, M. P. Recht, D. K. Sodickson, T. Pock, and F. Knoll. "Learning a Variational Network for Reconstruction of Accelerated MRI Data". In: *Magnetic Resonance in Medicine* 79.6 (2018), pp. 3055–3071. DOI: 10.1002/mrm.26977. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mrm.26977 (cit. on p. 8).

[37] R. Heckel and P. Hand. *Deep Decoder: Concise Image Representations from Untrained Non-convolutional Networks*. 2019. arXiv: 1810.03982 [cs.CV]. Pre-published (cit. on p. 52).

[38] K. Honauer, O. Johannsen, D. Kondermann, and B. Goldluecke. "A dataset and evaluation methodology for depth estimation on 4D light fields". In: *Asian Conference on Computer Vision*. Springer. 2016, pp. 19–34 (cit. on p. 37).

[39] Y. Hong, R. Kwitt, N. Singh, N. Vasconcelos, and M. Niethammer. "Parametric regression on the Grassmannian". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 38.11 (2016), pp. 2284–2297 (cit. on p. 62).

[40] M. Hosseini Kamal, B. Heshmat, R. Raskar, P. Vandergheynst, and G. Wetzstein. "Tensor Low-Rank and Sparse Light Field Photography". In: *Computer Vision and Image Understanding* 145 (2016). Light Field for Computer Vision, pp. 172–181. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2015.11.004 (cit. on pp. 23, 24).

[41] O. Johannsen, A. Sulc, and B. Goldluecke. "What Sparse Light Field Coding Reveals About Scene Structure". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 3262–3270. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.355 (cit. on p. 23).

[42] N. K. Kalantari, T.-C. Wang, and R. Ramamoorthi. "Learning-Based View Synthesis for Light Field Cameras". In: *ACM Transactions on Graphics* 35.6 (Nov. 2016), p. 193. DOI: 10.1145/2980179.2980251 (cit. on p. 23).

[43] D. Kim and J. A. Fessler. "Adaptive Restart of the Optimized Gradient Method for Convex Optimization". In: *Journal of Optimization Theory and Applications* 178.1 (July 2018), pp. 240–63. DOI: 10.1007/s10957-018-1287-4 (cit. on p. 53).

[44] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2015. arXiv: 1412.6980 (cit. on p. 53).

[45] A. Lahiri, G. Wang, S. Ravishankar, and J. A. Fessler. "Blind primed supervised (BLIPS) learning for MR image reconstruction". In: *IEEE Trans. Med. Imag.* 40.11 (Nov. 2021), 3113–24. DOI: 10.1109/TMI.2021.3093770 (cit. on p. 40).

[46] P. Lai, M. Lustig, S. S. Vasanawala, and A. C. S. Brau. "ESPIRiT (Efficient eigenvector-based L1SPIRiT) for compressed sensing parallel imaging - theoretical interpretation and improved robustness for overlapped FOV prescription". In: *Proc. Intl. Soc. Mag. Res. Med.* 2011, p. 65 (cit. on pp. 43, 44, 97).

[47] D. E. Lake and D. Keenan. "Maximum likelihood estimation of geodesic subspace trajectories using approximate methods and stochastic optimization". In: *IEEE Signal Process. Workshop Statis. Signal and Array Process.* IEEE. 1998, pp. 148–151 (cit. on pp. 59, 62, 63).

[48] M. Le Pendu, C. Guillemot, and A. Smolic. "A Fourier Disparity Layer Representation for Light Fields". In: *IEEE Transactions on Image Processing* 28.11 (Nov. 2019), pp. 5740–5753. ISSN: 1941-0042. DOI: 10.1109/TIP.2019.2922099 (cit. on p. 23).

[49] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proc. IEEE* 86.11 (Nov. 1998), 2278–2324. DOI: 10.1109/5.726791 (cit. on p. 55).

[50] A. Levin and F. Durand. "Linear View Synthesis Using a Dimensionality Gap Light Field Prior". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2010, pp. 1831–8. DOI: 10.1109/CVPR.2010.5539854 (cit. on pp. 12, 23, 29).

[51] A. Levin, W. T. Freeman, and F. Durand. "Understanding Camera Trade-Offs Through a Bayesian Analysis of Light Field Projections". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2008, pp. 88–101. DOI: 10.1007/978-3-540-88693-8_7 (cit. on p. 23).

[52] L. Li, W. Huang, I. Y.-H. Gu, and Q. Tian. "Statistical modeling of complex backgrounds for foreground object detection". In: *IEEE Trans. Image Process.* 13.11 (2004), pp. 1459–1472 (cit. on pp. 75, 97).

[53] K. Marwah, G. Wetzstein, Y. Bando, and R. Raskar. "Compressive Light Field Photography Using Overcomplete Dictionaries and Optimized Projections". In: *ACM Transactions on Graphics* 32.4 (July 2013), 46:1–12. DOI: 10.1145/2461912.2461914 (cit. on pp. 13, 23, 24).

[54] Y. Miyagi, K. Takahashi, M. P. Tehrani, and T. Fujii. "Reconstruction of Compressively Sampled Light Fields Using a Weighted 4D-DCT Basis". In: *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. IEEE, Sept. 2015, pp. 502–506. ISBN: 978-1-4799-8339-1. DOI: 10.1109/ICIP.2015.7350849 (cit. on pp. 23, 24, 34, 36).

[55] A. Mousnier, E. Vural, and C. Guillemot. *Partial Light Field Tomographic Reconstruction From a Fixed-Camera Focal Stack*. 2015. arXiv: 1503.01903 (cit. on p. 23).

[56] O. Nabati, R. Giryes, and D. Mendlovic. "Fast and Accurate Reconstruction of Compressed Color Light Field". In: *Proceedings of the IEEE International Conference Computational Photography (ICCP)*. May 2018, pp. 1–11. DOI: 10.1109/ICCPHOT.2018.8368477 (cit. on pp. 23, 24).

[57] P. Narayanamurthy and N. Vaswani. "Provable dynamic robust PCA or robust subspace tracking". In: *IEEE Trans. Inform. Theory* 65.3 (2018), pp. 1547–1577 (cit. on pp. 61, 62).

[58] G. Nataraj and R. Otazo. "Investigating Robustness to Unseen Pathologies in Model-Free Deep Multicoil Reconstruction". In: *Proceedings of the 2020 ISMRM Workshop on Sampling and Reconstruction*. 2020 (cit. on p. 8).

[59] R. Ng. "Fourier Slice Photography". In: *ACM Transactions on Graphics* 24.3 (July 2005), pp. 735–44. DOI: 10.1145/1073204.1073256 (cit. on p. 29).

[60] H. Nien. "Model-Based X-Ray CT Image and Light Field Reconstruction Using Variable Splitting Methods". PhD thesis. Ann Arbor, MI: Univ. of Michigan, Ann Arbor, MI, 48109-2122, 2014 (cit. on pp. 87, 89).

[61] R. Otazo, E. Candès, and D. K. Sodickson. "Low-rank plus sparse matrix decomposition for accelerated dynamic MRI with separation of background and dynamic

components". In: *Magnetic Resonance in Medicine* 73.3 (2015), pp. 1125–1136 (cit. on p. 59).

[62] L. Pfister and Y. Bresler. "Learning Filter Bank Sparsifying Transforms". In: *IEEE Transactions on Signal Processing* 67.2 (Jan. 2019), pp. 504–19. DOI: 10.1109/tsp.2018.2883021 (cit. on pp. 15, 17, 20, 32).

[63] A. Radford, L. Metz, and S. Chintala. "Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2016. arXiv: 1511.06434 (cit. on pp. 51, 55).

[64] A. Raj, Y. Li, and Y. Bresler. "GAN-Based Projector for Faster Recovery With Convergence Guarantees in Linear Inverse Problems". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019, pp. 5601–5610. DOI: 10.1109/ICCV.2019.00570 (cit. on p. 51).

[65] S. Ravishankar. "Adaptive sparse representations and their applications". PhD thesis. Univ. of Illinois, 2014 (cit. on p. 2).

[66] S. Ravishankar and Y. Bresler. "Learning Sparsifying Transforms". In: *IEEE Transactions on Signal Processing* 61.5 (Mar. 2013), pp. 1072–86. DOI: 10.1109/TSP.2012.2226449 (cit. on pp. 17, 24).

[67] S. Ravishankar and Y. Bresler. "Efficient Blind Compressed Sensing Using Sparsifying Transforms With Convergence Guarantees and Application to MRI". In: *SIAM Journal on Imaging Sciences* 8.4 (2015), pp. 2519–57. DOI: 10.1137/141002293 (cit. on pp. 17, 18, 21, 24, 42, 43, 46, 82).

[68] S. Ravishankar, A. Lahiri, C. Blocker, and J. A. Fessler. "Deep Dictionary-Transform Learning for Image Reconstruction". In: *Proceedings of the IEEE International Symposium on Biomedical Imaging (ISBI)*. Apr. 2018, pp. 1208–1212. DOI: 10.1109/ISBI.2018.8363788 (cit. on pp. 8, 15, 20).

[69] R. Rubinstein, T. Peleg, and M. Elad. "Analysis K-SVD: A dictionary-learning algorithm for the analysis sparse model". In: *IEEE Trans. Sig. Proc.* 61.3 (Feb. 2013), 661–77. DOI: 10.1109/TSP.2012.2226445 (cit. on p. 20).

[70] J. Schlemper, J. Caballero, J. V. Hajnal, A. N. Price, and D. Rueckert. "A Deep Cascade of Convolutional Neural Networks for Dynamic MR Image Reconstruction". In: *IEEE Transactions on Medical Imaging* 37.2 (Feb. 2018), pp. 491–503. ISSN: 1558-254X. DOI: 10.1109/TMI.2017.2760978 (cit. on p. 8).

[71] P. H. Schonemann. "A generalized solution of the orthogonal Procrustes problem". In: *Psychometrika* 31.1 (Mar. 1966), 1–10. DOI: 10.1007/BF02289451 (cit. on p. 14).

[72] V. Shah and C. Hegde. “Solving Linear Inverse Problems Using Gan Priors: An Algorithm With Provable Guarantees”. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Sept. 2018, pp. 4609–4613. ISBN: 9781538646588. DOI: 10.1109/ICASSP.2018.8462233 (cit. on pp. 51, 56).

[73] L. Shi, H. Hassanieh, A. Davis, D. Katabi, and F. Durand. “Light Field Reconstruction Using Sparsity in the Continuous Fourier Domain”. In: *ACM Transactions on Graphics* 34.1 (Dec. 2014), 12:1–13. DOI: 10.1145/2682631 (cit. on p. 23).

[74] A. Srivastava and E. Klassen. “Bayesian and geometric subspace tracking”. In: *Advances in Applied Probability* 36.1 (2004), pp. 43–56 (cit. on pp. 59, 62, 63).

[75] *Statistical Modeling of Complex Background for Foreground Object Detection: Demo Videos*. The Internet Archive. URL: https://web.archive.org/web/20080118111318/http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html (visited on 06/07/2022) (cit. on p. 97).

[76] Y. Sun, P. Babu, and D. P. Palomar. “Majorization-minimization algorithms in signal processing, communications, and machine learning”. In: *IEEE Trans. Sig. Proc.* 65.3 (Feb. 2017), 794–816. DOI: 10.1109/tsp.2016.2601299 (cit. on p. 66).

[77] K. Takahashi, S. Fujita, and T. Fujii. “Good Group Sparsity Prior for Light Field Interpolation”. In: *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. Sept. 2017, pp. 1447–1451. DOI: 10.1109/ICIP.2017.8296521 (cit. on p. 23).

[78] A. B. Taylor, J. M. Hendrickx, and F. Glineur. “Exact Worst-Case Performance of First-Order Methods for Composite Convex Optimization”. In: *SIAM Journal on Optimization* 27.3 (2017), pp. 1283–1313. DOI: 10.1137/16M108104X (cit. on p. 53).

[79] *The (New) Stanford Light Field Archive*. URL: http://lightfield.stanford.edu/lfs.html (visited on 07/23/2019) (cit. on pp. 10, 26).

[80] L. Thesing, V. Antun, and A. C. Hansen. *What Do AI Algorithms Actually Learn? - On False Structures in Deep Learning*. 2019. arXiv: 1906.01478 [stat.ML]. Pre-published (cit. on p. 8).

[81] M. Uecker, P. Lai, M. J. Murphy, P. Virtue, M. Elad, J. M. Pauly, S. S. Vasanawala, and M. Lustig. “ESPIRiT-an eigenvalue approach to autocalibrating parallel MRI: Where SENSE meets GRAPPA”. In: *Mag. Res. Med.* 71.3 (Mar. 2014), 990–1001. DOI: 10.1002/mrm.24751 (cit. on p. 97).

[82] D. Ulyanov, A. Vedaldi, and V. Lempitsky. *Deep Image Prior*. 2017. arXiv: 1711.10925 [stat.ML]. Pre-published (cit. on pp. 2, 52).

[83] S. Vagharshakyan, R. Bregovic, and A. Gotchev. "Accelerated Shearlet-Domain Light Field Reconstruction". In: *IEEE Journal of Selected Topics in Signal Processing* 11.7 (Oct. 2017), pp. 1082–1091. ISSN: 1932-4553. DOI: 10.1109/JSTSP.2017.2738617 (cit. on pp. 23, 24).

[84] S. Vagharshakyan, R. Bregovic, and A. Gotchev. "Light Field Reconstruction Using Shearlet Transform". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.1 (Jan. 2018), pp. 133–147. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2017.2653101 (cit. on pp. 23, 24).

[85] N. Vaswani, T. Bouwmans, S. Javed, and P. Narayanamurthy. "Robust subspace learning: Robust PCA, robust subspace tracking, and robust subspace recovery". In: *IEEE Signal Process. Mag.* 35.4 (2018), pp. 32–55 (cit. on pp. 59, 62).

[86] B. Wen, Y. Li, L. Pfister, and Y. Bresler. "Joint Adaptive Sparsity and Low-Rankness on the Fly: An Online Tensor Reconstruction Scheme for Video Denoising". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2017, pp. 241–250. DOI: 10.1109/ICCV.2017.35 (cit. on p. 33).

[87] B. Wen, S. Ravishankar, and Y. Bresler. "Video Denoising by Online 3D Sparsifying Transform Learning". In: *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. Sept. 2015, pp. 118–122. ISBN: 978-1-4799-8339-1. DOI: 10.1109/ICIP.2015.7350771 (cit. on pp. 17, 24).

[88] B. Wilburn, N. Joshi, V. Vaish, E.-V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy. "High Performance Imaging Using Large Camera Arrays". In: *ACM Transactions on Graphics* 24.3 (July 2005), pp. 765–776. ISSN: 0730-0301. DOI: 10.1145/1073204.1073259 (cit. on p. 12).

[89] G. Wu, M. Zhao, L. Wang, Q. Dai, T. Chai, and Y. Liu. "Light Field Reconstruction Using Deep Convolutional Network on EPI". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017, pp. 1638–1646. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.178 (cit. on pp. 23, 24).

[90] B. Yang. "Projection approximation subspace tracking". In: *IEEE Trans. Signal process.* 43.1 (1995), pp. 95–107 (cit. on pp. 59, 62).

[91] Y. Yoon, H.-G. Jeon, D. Yoo, J.-Y. Lee, and I. S. Kweon. "Light-Field Image Super-Resolution Using Convolutional Neural Network". In: *IEEE Signal Processing Letters* 24.6 (June 2017), pp. 848–852. ISSN: 1070-9908. DOI: 10.1109/LSP.2017.2669333 (cit. on p. 23).

[92] J. Zbontar, F. Knoll, A. Sriram, M. J. Muckley, M. Bruno, A. Defazio, M. Parente, K. J. Geras, J. Katsnelson, H. Chandarana, Z. Zhang, M. Drozdzal, A. Romero, M. Rabbat, P. Vincent, J. Pinkerton, D. Wang, N. Yakubova, E. Owens, C. L. Zitnick, M. P.

Recht, D. K. Sodickson, and Y. W. Lui. *fastMRI: An Open Dataset and Benchmarks for Accelerated MRI*. 2018. arXiv: 1811.08839. Pre-published (cit. on pp. 44, 85).

[93] D. Zhang, Z. Xu, Z. Huang, A. R. Gutierrez, I. Y. Chun, C. J. Blocker, G. Cheng, Z. Liu, J. A. Fessler, Z. Zhong, and T. B. Norris. "Graphene-Based Transparent Photodetector Array for Multiplane Imaging". In: *Conference on Lasers and Electro-Optics (CLEO)*. Optical Society of America, 2019. DOI: 10.23919/CLEO.2019.8750012.

[94] D. Zhang, Z. Xu, Z. Huang, A. R. Gutierrez, C. J. Blocker, C.-H. Liu, M.-B. Lien, G. Cheng, Z. Liu, I. Y. Chun, J. A. Fessler, Z. Zhong, and T. B. Norris. "Neural network based 3D tracking with a graphene transparent focal stack imaging system". In: *Nature Communications* 12.1 (2021), pp. 1–7.

[95] S. Zhang, Y. Lin, and H. Sheng. "Residual Networks for Light Field Image Super-Resolution". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 11038–11047. DOI: 10.1109/CVPR.2019.01130 (cit. on p. 23).

[96] H. Zhao, O. Gallo, I. Frosio, and J. Kautz. "Loss Functions for Image Restoration With Neural Networks". In: *IEEE Transactions on Computational Imaging* 3.1 (Mar. 2017), pp. 47–57. ISSN: 2333-9403. DOI: 10.1109/TCI.2016.2644865 (cit. on p. 53).

[97] X. Zheng, S. Ravishankar, Y. Long, and J. A. Fessler. "PWLS-ULTRA: An Efficient Clustering and Learning-Based Approach for Low-Dose 3D CT Image Reconstruction". In: *IEEE Transactions on Medical Imaging* 37.6 (June 2018), pp. 1498–510. DOI: 10.1109/TMI.2018.2832007 (cit. on p. 32).