IMAGE RESTORATION FROM NOISY AND LIMITED MEASUREMENTS WITH APPLICATIONS IN 3D IMAGING

BY

HUY QUANG BUI

DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical and Computer Engineering in the Graduate College of the University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Doctoral Committee:
Professor Minh Do, Chair
Professor Yoram Bresler
Professor John Hart
Professor Thomas Huang
Professor Andrew Singer
Abstract

The recovery of image data from noisy and limited measurements is an important problem in image processing with many practical applications. Despite great improvements in imaging devices over the past few years, the need for a fast and robust recovery method is still essential, especially in fields such as medical imaging or remote sensing. These methods are also important for new imaging modalities where the quality of data is still limited due to current state of technology.

This thesis investigates novel methods to recover signals and images from noisy or sparse measurements, in new imaging modalities, for practical 3D imaging applications. In particular, the following problems are considered.

First, the Tree-based Orthogonal Matching Pursuit (TOMP) algorithm is proposed to recover sparse signals with tree structure. This is an improvement over the Orthogonal Matching Pursuit method with the incorporation of the sparse-tree prior on the data. A theoretical condition on the recovery performance as well as a detailed complexity analysis is derived. Extensive experiments are carried out to compare the proposed method with other state-of-the-art algorithms.

Second, a new point clouds registration method is investigated and applied for 3D model reconstruction with a depth camera, which is a recently introduced device with many potential applications in 3D imaging and human-machine interaction. Currently, the depth camera is limited in resolution and suffers from complex types of noise. In the proposed method, the Implicit Moving Least Squares (IMLS) method is employed to derive a more robust registration method which can deal with noisy point clouds. Given a good registration, information from multiple depth images can be integrated together to help reduce the effects of noise and possibly increase the resolution. This method is essential to bring commodity depth cameras to new applications that demand accurate depth information.
Third, a hybrid system which consists of a light-field camera and a depth camera rigidly attached together is proposed. The system can be applied for digital refocusing on an arbitrary surface and for recovering complex reflectance information of a surface. The light-field camera is a device that can sample the 4D spatio-angular light field and allows one to refocus the captured image digitally. Given light-field information, it is possible to rearrange the light rays appropriately to render novel views or to generate refocused photographs. In theory, it is possible to estimate the depth map from a light field. However, there is a trade-off between angular and spatial resolution in current designs of light-field cameras, which leads to low quality and resolution of the estimated depth map. Moreover, for advanced 3D imaging applications, it is important to have good quality geometric and radiometric information. Thus, a depth camera is attached to the light-field camera to achieve this goal. The calibration of the system is presented in detail. The proposed system is demonstrated to create a refocused image on an arbitrary surface. However, we believe that the proposed system has great potential in more advanced imaging applications.
To my grandparents and my parents
and my small family
Acknowledgments

My deepest gratitude goes to my adviser Minh Do for his continuous support and encouragement during my study. His enthusiasm about technology gives me great motivation for my research. I would like to thank my many colleagues for their beneficial discussions and interactions, especially Joshua Blackburn, Daniel Kubacki, Quang Nguyen, Hien Nguyen, Ramanpreet Singh Pahwa, Andre Targino, Benjamin Chidester, Gregory Meyer, Siying Liu, Ha Nguyen, Trong Nguyen, Tan Nguyen, Kerui Min, Huixuan Wang, Chen Chen and Raymond Yeh.

Additionally, I would like to thank Professor Yoram Bresler, Professor John Hart, Professor Thomas Huang and Professor Andrew Singer for serving in my PhD committee and for their invaluable advice and feedback.

I also want to thank my friends at UIUC for the great interactions during the time of my study. I want to thank my grandparents, my parents, my aunts and many other relatives for their years of love and support. Finally, I would like to thank my wife Ha for her sacrifice and understanding.
# Table of Contents

## Chapter 1 Introduction
- 1.1 Motivations ........................................... 1
- 1.2 Problem Statement .................................... 5
- 1.3 Thesis Contributions and Summary ............... 5

## Chapter 2 A Tree-Based Algorithm to Recover Sparse-Tree Signal
- 2.1 Introduction ........................................... 7
- 2.2 Background ........................................... 8
- 2.3 The Sparse-Tree Model of Signals ................. 10
- 2.4 Tree-based Orthogonal Matching Pursuit Algorithm ........................................... 13
- 2.5 Implementation Details ............................. 15
- 2.6 Analysis of TOMP ..................................... 16
- 2.7 Experimental Results ............................... 25
- 2.8 Discussion ............................................. 30
- 2.9 Concluding Remarks ................................. 32

## Chapter 3 A New Method for Registration of 3D Point Clouds
- 3.1 Introduction ........................................... 34
- 3.2 Problem Statement ................................... 35
- 3.3 Related Work ......................................... 36
- 3.4 Background .......................................... 38
- 3.5 Proposed Algorithm ................................ 42
- 3.6 Analysis and Comparison .......................... 47
- 3.7 Application in 3D Reconstruction .................. 52
- 3.8 Experimental Results ............................... 56
- 3.9 Concluding Remarks ................................. 60

## Chapter 4 A Light-Field and Depth Camera System and Applications
- 4.1 Introduction ........................................... 64
- 4.2 Related Work ......................................... 65
- 4.3 Background .......................................... 68
- 4.4 System Setup and Calibration ...................... 71
- 4.5 Refocusing on an Arbitrary Surface ................ 73
- 4.6 Potential Applications of the Proposed System ................ 80
- 4.7 Experimental Results and Discussions ............ 81
4.8 Concluding Remarks ........................................ 82
Chapter 5 Conclusion ............................................. 87
Appendix A Additional Refocused Results using Our Proposed System 88
References .......................................................... 93
Chapter 1
Introduction

1.1 Motivations

The recovery of image data from noisy and limited measurements is very important in many practical imaging applications. Despite great improvements in imaging devices over the past few years, the need for fast and robust recovery methods is still vital for many applications. Moreover, new image modalities and high-dimensional image data call for novel recovery algorithms that can handle them effectively.

In medical imaging, it is highly desirable to have high-quality and high-resolution images for accurate diagnostic and detection. For many medical imaging modalities, it may require a long time to acquire such data and the patient must remain still during the capturing time, which is hard to achieve without discomfort. To reduce acquisition time, the captured data are usually sparse and have low quality. Thus, good methods to recover the image from sparse samples and make it useful for later diagnosis are highly desired.

The introduction of compressed sensing [1, 2, 3, 4, 5, 6] has partly answered the above need. The key idea in compressed sensing is that sparse signals can be recovered from a small number of random and non-adaptive measurements, far below the Nyquist sampling rate. So far, most researches in compressed sensing have been focusing on the sparsity of the signals to be recovered and have not paid much attention to special structures embedded in the signals. In practice, many signals exhibit some special structures such as block sparsity or tree sparsity. This observation has led to the concept of model-based compressed sensing [7]. It has been shown that exploiting special structures in the signals can improve the recovery significantly.

Recently, several new devices have been introduced that can capture new
image modalities in faster and cheaper ways than ever and bring more interesting applications to any users. These devices include the commodity depth camera and the light-field (or plenoptic) camera.

![Kinect 1.0](image1) ![Kinect 2.0](image2)

Figure 1.1: The Kinect cameras from Microsoft. It was introduced as a sensor for the XBox 360 gaming console. A version for PC, named Kinect for PC, is available for developer with an supported software development kit, the Kinect SDK.

Commodity depth cameras, such as the Kinect camera in Figure 1.1, offer the ability to capture depth information and create a revolution in 3D imaging. Before the invention of the depth camera, people used to capture 3D information with slower, more expensive devices or more complicated techniques. With the invention of the depth camera, 3D information can be captured at up to 30 frames per second. This capability is very useful for interactive 3D imaging applications. Since current depth cameras can only capture depth maps at low resolution and acceptable quality, a good method to reconstruct a high-quality 3D scene from a sequence of low-quality depth images is very important for practical applications. Figure 1.2 shows the typical depth image from a time-of-flight camera versus a structured-light camera.

In 3D imaging, due to the limited field of view of the capturing devices and self-occlusion, multiple views of the objects are captured and then merged together to generate the complete 3D model. Before merging, data from different viewpoints need to be registered together. Point cloud registration is an active research field with many methods proposed in the last two decades. Iterative Closest Point (ICP) \[8, 9, 10\] is the most popular method for rigid registration of two point sets, assuming a rough initial alignment. Many variants have been proposed to improve the performance of ICP \[11\]. These ICP-based methods work well on clean point clouds, such as those given by
Figure 1.2: The depth maps from a time-of-flight (ToF) and a structured-light (SL) camera, which demonstrate different noise characteristics. ToF depth images are affected by severe noise and a systematic bias whereas SL depth maps have missing data.

Another new device, the light-field camera, can capture 4D spatio-angular light-field and allows the captured images to be refocused digitally. The first commercial light-field camera, the Lytro camera, allows one to refocus an image after capture and attracts a lot of interest from the photography community. However, the first version of Lytro was disappointing due to
the low resolution of the rendered images. A new version of the camera, the Lytro Illum, was recently introduced to overcome the limitation but at higher cost. Figure 1.3 shows the two versions of the Lytro camera. The light-field camera has great potential in applications where one would like to capture the whole light-field in front of the camera. For example, by refocusing on an arbitrary surface, it could be applied for touchless scanning of a document on an arbitrary surface, which is very useful for digitizing historical documents or scanning thick book without distortion. Recently, Fujitsu introduced a contactless scanner, the Fujitsu ScanSnap SV600 shown in Figure 1.4 which can scan document without contact. With a light-field camera, it is quite possible to develop a portable touchless scanner for personal use on the go. Another possible application is capturing a surface light-field, which is the collection of rays emitted or reflected from a surface. Given the surface light-field, it is possible to render virtual images of an object under novel viewing directions. Moreover, if the light source can be controlled, then the reflectance property of the surface can also be estimated. The reflectance property, together with geometric information, is essential to render a 3D object realistically under novel viewpoints and lighting conditions. Depth and light-field cameras are effective tools to access this information.

In summary, the main motivation in this thesis is new methods to recover data from noisy and sparse measurements, which can exploit the special structures embedded in the data. First, a tree-based recovery algorithm to recover sparse signal with tree structure will be presented. Next, a novel registration method for noisy point clouds from depth cameras and its appli-
cations to 3D imaging will be developed. Finally, a novel system consisting of depth and light-field cameras will be proposed and its applicability for complex refocusing problems and surface light-field capturing will be demonstrated.

1.2 Problem Statement

This thesis investigates new methods for signal and image recovery from sparse and noisy measurements. These methods should be able to exploit special structures and properties of the data. In particular, the following problems will be addressed:

1. Investigate a novel method to recover a sparse signal with tree structure.

2. Develop a new point cloud registration method that is robust with noisy depth data from commodity depth camera and extend it for 3D reconstruction.

3. Develop novel applications that exploit both geometric and radiometric information, captured by a depth camera and a light-field camera.

1.3 Thesis Contributions and Summary

Chapter 2 will present the Tree-based Orthogonal Matching Pursuit (TOMP) algorithm, a simple and fast sparse recovery algorithm for signals having sparse-tree structure. Theoretical analysis of the proposed algorithm will be provided in detail, together with experimental results comparing TOMP with other methods.

Chapter 3 will focus on a novel point-cloud registration algorithm, named the Signed Distance Registration (SDR), and its application in 3D reconstruction. The algorithm will be analyzed based on 2D toy examples, together with experimental results on 3D data. We will propose a 3D reconstruction method using SDR as a building block.

Chapter 4 will propose a new imaging system consisting of a light-field camera and a depth camera. The system will be demonstrated for digital
refocusing on arbitrary surface. We will show that the system can be applied to a wide range of imaging applications with use light-field and depth as input.
Chapter 2

A Tree-Based Algorithm to Recover Sparse-Tree Signal

2.1 Introduction

In compressed sensing, the existence of a sparse representation of an unknown signal in a certain basis has been used extensively as prior knowledge for signal reconstruction from a limited number of measurements [1, 2, 3, 4, 5, 6]. Recently, several recovery algorithms have incorporated special signal structures such as block sparsity or tree sparsity into the reconstruction process [7, 12, 13, 14, 15].

In [15, 16] the Tree-based Orthogonal Matching Pursuit (TOMP) algorithm was primitively introduced to recover signals with a sparse-tree prior. In this chapter, we further investigate TOMP and derive a sufficient condition for successful recovery of sparse-tree signals. Moreover, more experiments are carried out to compare TOMP with more recent tree-based recovery methods. The results show that TOMP provides competitive quality with more sophisticated methods, while being much simpler.

The main contributions of this chapter are new results on TOMP, including:

- A new theoretical analysis of the algorithm’s performance,
- More organized experiments comparing the proposed algorithm with other tree-based reconstruction algorithms,
- A detailed analysis on computational complexity of TOMP.

The outline of the chapter is as follows. First, existing sparse inverse problems and reconstruction algorithms are reviewed in Section 2.2. After that, the sparse-tree model will be presented in Section 2.3. The proposed TOMP
2.2 Background

2.2.1 Sparse Inverse Problem and Compressed Sensing

For an unknown signal \( s \) of length \( N \), suppose that only a limited number of non-adaptive linear measurements \( (M \ll N) \) can be acquired due to physical constraints:

\[
\Phi s = b, \tag{2.1}
\]

where \( \Phi \) is a fixed \( M \times N \) measurement matrix, and \( b \) is a length-\( M \) vector that contains the measured data. The inverse problem is to reconstruct signal \( s \) from \( b \).

Suppose that \( s \) has an expansion in some basis via a fixed \( N \times N \) transform matrix \( W \) as

\[
s = Wx. \tag{2.2}
\]

If only \( K \) out of \( N \) entries \( (K \ll N) \) of \( x \) are non-zero, then \( x \) is called a \( K \)-sparse signal. The number of non-zero coefficients in \( x \) is called the sparsity of \( x \):

\[
S(x) = \|x\|_0 = \text{size}\{i : x_i \neq 0\}. \tag{2.3}
\]

Let \( A = \Phi W \), then the inverse problem (2.1) becomes

\[
Ax = b. \tag{2.4}
\]

Since \( M \ll N \), both (2.1) and (2.4) are underdetermined systems. To solve (2.4), most current methods use the sparse prior and search for the sparsest solution:

\[
\min_x S(x) \quad \text{s.t.} \quad Ax = b. \tag{2.5}
\]

We refer to (2.5) as the sparse inverse problem.
In practice, most signals are not exactly sparse but can be well approximated by sparse signals [17]. In such signals, only $K \ll N$ coefficients have significant values (i.e. greater than some threshold) while the remaining coefficients have very small values; and when these coefficients are sorted in descending order, the coefficient magnitude decays quickly. These signals are called compressible signals. All compressed sensing algorithms work with compressible signals.

2.2.2 Existing Sparse Reconstruction Algorithms

Problem (2.5) is an NP-hard problem. One approach to solve (2.5) is to use greedy search methods, such as [18, 6, 19, 20, 21], to find the indices of significant coefficients in $x$. A popular and simple greedy method is Orthogonal Matching Pursuit (OMP) [18, 6]. Originally, OMP was developed to find the optimal sparse representations of a signal in a redundant dictionary. Each OMP iteration searches the dictionary for the atom that is most correlated with the residual from the previous iteration and estimates the value of the corresponding coefficient by orthogonally projecting the data onto the whole set of selected atoms. The main limitation of OMP is that each iteration must correctly select an index. Once an index has been selected, it cannot be removed from the selected set. StOMP [21], CoSaMP [19], and Subspace Pursuit [20] improve upon OMP by selecting more than one index at each iteration and including a backtracking or pruning step to refine the selected set.

Besides the greedy approaches, there are many other approaches to solve the sparse inverse problem including Basic Pursuit or $l_1$-minimization [22, 3, 4, 5], Majorization-Minimization [23, 24, 25] and Approximate Message Passing (AMP) [26].

2.2.3 Model-Based Reconstruction Algorithms

Recently, more research has been focused on recovery of sparse signals that embed additional structures. Intuitively, taking into account these structures makes the recovery process easier, in terms of fewer required measurements or faster recovery time.
In 2005, [14] and [15] independently proposed to exploit the sparse-tree structure in the wavelet transform of piecewise smooth signals in compressed sensing. In [15, 16] TOMP was introduced to recover signals with a sparse-tree prior. The same structure was also exploited in the Tree-based Majorization-Minimization (TMM) algorithm [27] which is an extension of Majorization-Minimization (MM) approach.

In 2010, [7] formulated the theory of Model-based Compressed Sensing for the recovery of sparse and compressible signals that have sparse-tree or sparse-block structures. These theories proved that it would be advantageous to use special signal structures as additional priors for signal recovery. In particular, the authors proposed two methods to recover sparse-tree and sparse-block signals based on CoSaMP [19] and Iterative Hard Thresholding (IHT) [28] algorithms.

The tree-structure of wavelet coefficients was also incorporated into Bayesian-based methods, such as [12] and [13]. In these papers, the tree-structure was embedded in a statistical model and different Bayesian inference methods were used for the reconstruction, such as Variational Bayesian [13] or Markov Chain Monte Carlo [12]. Although these methods can give high-quality results, the main disadvantage is their expensive computation, as will be discussed later. Recently, [29] proposed a new message-passing-based method for compressive imaging that takes into account the Markov-tree prior.

2.3 The Sparse-Tree Model of Signals

2.3.1 The Sparse-Tree Model

Many signals that we encounter in practice can be modelled as a tree. In this chapter, we consider the following sparse-tree model.

**Definition 1** (Sparse-tree model). A signal $\mathbf{x}$ is said to conform to the sparse-tree model if it satisfies the following properties:

1. $\mathbf{x}$ is sparse or compressible.
2. The coefficients of $\mathbf{x}$ can be organized into one or many trees.
3. The non-zero or significant coefficients of $\mathbf{x}$ are connected together in rooted sub trees of the trees.

4. When going from the roots to the leaves of the trees, the maximum magnitude of the coefficients at each level will be decreasing.

This sparse-tree model might seem very restrictive at the first glance. However, it is an effective model for many real world problems and signals. In the following section, we provide two examples in practice where this model can be applied.

### 2.3.2 Examples of the Sparse-Tree Model

**Example 1**

An interesting problem in which the sparse-tree model can be applied is modelling the spreading of a disease in a population. In this type of problem, each person in that population is represented by a vertex of the tree and the sources of the disease are represented by the roots. Suppose that we do some random group testing and we are interested in discovering people in that population who have acquired the disease and the seriousness of the disease exhibited at each of them. In this case, $\mathbf{x}$ is the vector of coefficients where each coefficient represents the seriousness of the disease manifesting in each person. Intuitively, the seriousness is decreasing with distance from the sources. A problem similar to this case is the passing of a rumor in a population where the roots are the sources of the rumor and the coefficients represent the confidence of the rumor.

**Example 2**

Another example of the sparse-tree model is in the wavelet coefficients of a piecewise smooth signal. Consider (2.2) when the transform $\mathbf{W}$ is an $L$-level 1-dimensional wavelet transform. In that case, the entries of $\mathbf{x}$ consist of

$$
\mathbf{x} = \left\{ \{x_{0,p}^{(s)}\}_{1 \leq p \leq N/2^L}, \{x_{i,p}^{(w)}\}_{1 \leq i \leq L, 1 \leq p \leq N/2^{L-i+1}} \right\},
$$
where $x_{0,p}^{(s)}$ are the scaling coefficients and $x_{1,p}^{(w)}$ are the wavelet coefficients. In this notation, $l$ is the scale of a wavelet coefficient, where 1 is the coarsest scale and $L$ is the finest scale, and $p$ is the shift of a coefficient in a level. The elements of $\mathbf{x}$ can be arranged into binary trees where the roots are the wavelet coefficients at the coarsest level, as in Figure 2.1. Each wavelet coefficient $x_{l,p}^{(w)}$ has two children $x_{l+1,2p-1}^{(w)}$ and $x_{l+1,2p}^{(w)}$. The entries in $\mathbf{x}$ can be specified either in the vector form $x_i$ or in the tree-structured form $x_{l,p}$.

![Figure 2.1: The tree structure of the coefficient vector resulted from a 3-level wavelet decomposition of a length-16 signal, where black nodes represent significant coefficients and white nodes represent insignificant coefficients.](image)

By examining the wavelet transform $\mathbf{x}$ of a piecewise-smooth signal, three distinguishing properties can be observed:

P1 Vector $\mathbf{x}$ is sparse or compressible; i.e. only a few entries in $\mathbf{x}$ are non-zero or significant.

P2 The non-zero or significant entries of $\mathbf{x}$ are likely connected in a tree structure.

P3 The wavelet coefficients tend to decay across scale from coarse to fine scales [30].

Properties P2 and P3 are important additional priors that have not been considered adequately in recovery algorithms. These properties hold because each discontinuity of the signal generates a set of large wavelet coefficients in a “cone of influence” [30], which is also referred to as the wavelet footprint [31]. In particular, if a coefficient is non-zero or significant then its ancestors are more likely to be non-zero or significant. Therefore, the significant coefficients of $\mathbf{x}$ themselves form the rooted sub-trees as illustrated in Figure 2.1.
2.3.3 Descendant and Ancestor

In this section, we present several important concepts that will facilitate the development of our proposed algorithm. These concepts were introduced in the context of wavelet tree model \[32\] and are recalled here for completeness. Consider a signal $\mathbf{x}$ whose coefficients can be organized into one or several trees.

A descendant of a node $x_i$ is a node $x_j$ that can be reached from node $x_i$ by following the children nodes. For example, in Figure 2.1, all descendants of node $x_1^{(w)}$ are $\{x_2^{(w)}, x_4^{(w)}, x_5^{(w)}\}$.

An ancestor of a node $x_i$ is a node $x_j$ that can be reached from $x_i$ by following the parent nodes. The history set of a node $x_i$ is the set of ancestors of $x_i$ up to its root. For example, in Figure 2.1, $x_1^{(w)}$ is an ancestor of node $x_3^{(w)}$, and the history set of node $x_3^{(w)}$ is $\{x_2^{(w)}, x_1^{(w)}\}$.

2.4 Tree-based Orthogonal Matching Pursuit Algorithm

This section describes in detail our proposed Tree-based Orthogonal Matching Pursuit (TOMP) algorithm for solving the sparse inverse problem for signals with the sparse-tree characteristic. Let $\mathbf{x}$ be a $K$-sparse signal in $\mathbb{R}^N$ whose non-zero coefficients are all connected in a binary tree. Let $\mathbf{A}$ be a $M \times N$ random measurement matrix and $\mathbf{b} = \mathbf{A}\mathbf{x}$ be the vector of measurements. The $i^{th}$ column of $\mathbf{A}$ is denoted $\mathbf{a}_i$.

Let $\Lambda_k$ be the set of selected indices after the $k^{th}$ iteration. The initial set $\Lambda_0$ consists of indices of entries which are expected to be significant at the root of the tree. For example, in the case of wavelet decomposition, all scaling coefficients are likely to be significant.

Algorithm 2 (TOMP for signal conforming to the sparse-tree model).

**INPUT**

- $M \times N$ measurement matrix $\mathbf{A}$.
- Length-$M$ vector $\mathbf{b}$ of measurements.
- Relaxation parameter $\alpha \in [0, 1]$. 

13
• Downward extending parameter $d$.
• Index array $P$ that maps each node index to the index of its parent.
• Index array $Q$ that contain the indices of the children of each node.

**OUTPUT**
• Reconstructed vector $\hat{x}$ of length $N$ which has a sparse-tree structure.

**PROCEDURE**
1. Initialize the set $\Lambda_0$ with indices of the entries at the roots of the trees.
   Initialize the residual as
   $$r_0 = b - P_{\text{span}\{a_i : i \in \Lambda_0\}} b.$$  \hfill (2.6)

   Set $k = 0$ and $\text{stop} = \text{false}$.

2. While $\text{stop} == \text{false}$:
   (a) $k = k + 1$.
   (b) Form the candidate set $C_k$ that contains the indices of the descendants of selected nodes within $d$ levels.
   $$C_k = \bigcup_{i \in \Lambda_{k-1}} D_d(i),$$ \hfill (2.7)

   where $D_d(i)$ denotes the index set of all descendants of nodes $x_i$ within $d$ levels. Hence $d$ is named the downward extending parameter.

   (c) Form the finalist set $F_k$ as
   $$F_k = \{i \in C_k \text{ s.t. } |\langle r_{k-1}, a_i \rangle| \geq \alpha \max_{j \in C_k} |\langle r_{k-1}, a_j \rangle|\},$$ \hfill (2.8)

   where $\alpha$ is a given relaxation parameter.

   (d) Select the index $i_k$ from the finalist set $F_k$ such that
   $$i_k = \arg \min_{i \in F_k} \|b - P_{\text{span}\{a_j : j \in \Lambda_{k-1} \cup H(i)\}} b\|_2,$$ \hfill (2.9)
where $H(i)$ denotes the index set of the ancestors of node $x_i$ up to the root.

(e) Augment the index set as

$$\Lambda_k = \Lambda_{k-1} \cup H(i_k),$$  

(2.10)

where $H(i_k)$ is the index set of the ancestors of node $x_{i_k}$ up to the roots.

(f) Update the residual

$$r_k = b - P_{\text{span}\{a_i : i \in \Lambda_k\}}b.$$  

(2.11)

(g) If $\|r_k\|_2^2 \leq \varepsilon$, a selected threshold, or the number of selected columns exceed a certain limit, e.g. $M/2$, set $\text{stop} = \text{true}$.

3. Non-zero coefficients of the estimated signal $\hat{x}$, indexed by $\Lambda_k$, are the solution of

$$A_{\Lambda_k} \hat{x}_{\Lambda_k} = b - r_k,$$  

(2.12)

where $A_{\Lambda_k}$ consists of columns of $A$ indexed by $\Lambda_k$.

2.5 Implementation Details

In this section, an implementation of Algorithm 2 based on Gram-Schmidt orthogonalization process is provided. In this implementation, whenever a new column $a_i$ is selected, $i$ is stored into the selected set $\Lambda_k$. At the same time, $a_i$ is orthonormalized with respect to all of the previously selected columns and then stored in the set of orthonormalized selected columns $U_k$, named the Gram-Schmidt selected set.

After equations (2.6)-(2.8), for each node $i \in F_k$, a corresponding sub-tree containing that node and its ancestors is formed. Each column in $\{a_{H(i)}\}$ is orthonormalized against $U_{k-1}$ and the remaining columns in $\{a_{H(i) \backslash \Lambda_{k-1}}\}$ to form the set of orthonormalized columns $\{a_{\overline{H(i)}}\}$. This step is performed by using the Gram-Schmidt process.

Then, the current residual is projected onto each sub-tree and the resulting
residuals are recorded and compared.

\[
\mathbf{r}_{\text{temp}} = \mathbf{b} - P_{\text{span}\{a_j : j \in \Lambda_{k-1} \cup H(i)\}} \mathbf{b} \\
= \mathbf{r}_{k-1} - \sum_{j \in H(i) \setminus \Lambda_{k-1}} \langle \mathbf{r}_{k-1}, a_j \rangle a_j. 
\]

The sub-tree that gives the smallest residual is selected.

\[
i_k = \arg \min_{i \in \mathcal{F}_k} \| \mathbf{r}_{\text{temp}} \|. 
\]

(2.13)

This gives the solution of (2.9). The selected set \( \Lambda_k \) and the new residual \( \mathbf{r}_k \) are updated through (2.10) and (2.11).

The selected set \( \mathcal{U}_k \) is updated by adding the selected orthonormalized sub-tree:

\[
\mathcal{U}_k = \mathcal{U}_{k-1} \cup \{ a_{H(i_k) \setminus \Lambda_{k-1}} \}. 
\]

(2.14)

The algorithm terminates when the stopping rules are satisfied. By caching the set of orthonormalized selected columns, the computational cost at each iteration is significantly reduced.

2.6 Analysis of TOMP

In this section, we provide a theoretical analysis of the performance of the proposed algorithm. First, we derive a sufficient condition for the recovery of sparse tree signal using TOMP in Section 2.6.1. Next, we analyze the effect of the parameters on the algorithm in Section 2.6.2. Finally, Section 2.6.3 analyzes the computational complexity of the proposed algorithm and compares it with other methods.

2.6.1 Reconstruction Condition

To set the context for our analysis, we recall the definition of a cumulative coherence function in [33].

**Definition 3** (Cumulative coherence function). Given a dictionary \( \mathcal{D} \) with atoms \( \phi_\lambda \). Let \( \Omega \) be the index set of all atoms in \( \mathcal{D} \).

16
The cumulative coherence function is defined as:

$$\mu_1(K) = \max_{\Lambda \subseteq \Omega, |\Lambda| = K} \max_{\Psi \in \Omega \setminus \Lambda} \sum_{\lambda \in \Lambda} |\langle \psi, \phi_\lambda \rangle|,$$

(2.15)

where $\Lambda$ is a subset of $\Omega$ and $\psi$ is an atom in the remaining set $\Omega \setminus \Lambda$.

The condition for reconstruction, as stated in Theorem 6, is the same as the condition for Orthogonal Matching Pursuit [33]. However, in this chapter, the condition is derived for recovery of sparse-tree signals using our proposed TOMP algorithm. For completeness, the result of [33] is restated here.

**Theorem 4** (Theorem 3.5 in [33]). Suppose that $\mu_1$ is the cumulative coherence function of $\mathcal{D}$. OMP will recover the sparsest $K$-term representation of the signal whenever

$$\mu_1(K - 1) + \mu_1(K) < 1.$$  

(2.16)

Before stating our results, we would like to introduce some basic notation required for later development. Let $x$ be the vector to be recovered and assume that $x$ satisfies the sparse-tree model. Let $\Lambda$ be the index set of non-zero coefficients of $x$, $|\Lambda| = K$. Nodes on the tree that have index in $\Lambda$ are called optimal nodes. The data vector $b$ can be represented as

$$b = \sum_{i \in \Lambda} x_i a_i.$$

If after the $(k-1)^{th}$ iteration, the algorithm has found correctly the indices of some non-zero coefficients in $x$, i.e. $\Lambda_{k-1} \subseteq \Lambda$, then the residual can be represented as

$$r_{k-1} = b - P_{\text{span}(a_i : i \in \Lambda_{k-1})} b$$

$$= \sum_{i \in \Lambda} x_i a_i - \sum_{i \in \Lambda_{k-1}} \beta_i^{(k-1)} a_i$$

$$= \sum_{i \in \Lambda \setminus \Lambda_{k-1}} \left(x_i - \beta_i^{(k-1)}\right) a_i + \sum_{i \in \Lambda \setminus \Lambda_{k-1}} x_i a_i$$

$$= \sum_{i \in \Lambda \setminus \Lambda_{k-1}} c_i^{(k-1)} a_i,$$
where
\[
c_i^{(k-1)} = \begin{cases} 
  x_i - \beta_i^{(k-1)}, & i \in \Lambda_{k-1} \\
  x_i, & i \notin \Lambda_{k-1}
\end{cases} \tag{2.17}
\]

Also, note that
\[
|\langle r_{k-1}, a_i \rangle| = 0 \quad \forall i \in \Lambda_{k-1}
\]

since \( r_{k-1} \) is the residual after orthogonally projecting \( b \) onto the space spanned by the columns \( \{a_i\}_{i \in \Lambda_{k-1}} \).

In the following analysis, the levels of the nodes on the tree are increasing from roots to leaves, with 1 being the level of the roots, as described in Figure 2.1.

**Lemma 5.** Assume that the signal \( x \) conforms to the sparse-tree model. Let \( \tilde{C}_k = \Lambda_{k-1} \cup C_k \), where \( C_k \) contains all the nodes descending from the nodes in \( \Lambda_{k-1} \) up to \( d \) levels. Then
\[
\max_{j \in \Lambda} |c_j^{(k-1)}| = \max_{j \in \Lambda \cap \tilde{C}_k} |c_j^{(k-1)}| \tag{2.18}
\]

or equivalently,
\[
\max_{j \in \Lambda \setminus \tilde{C}_k} |c_j^{(k-1)}| \leq \max_{j \in \Lambda \cap \tilde{C}_k} |c_j^{(k-1)}|. \tag{2.19}
\]

*Proof.* Suppose that \( x \) is a signal which conforms to the sparse-tree model defined in Section 2.3. Let \( L_n \) be the set of indices at level \( n \) of the tree. Then
\[
\max_{i \in L_n} |x_i| = \max_{i \in \Lambda \cap L_n} |x_i|, \tag{2.20}
\]
since \( |x_i| = 0 \quad \forall i \notin \Lambda \).

At the \( k \text{th} \) iteration, the candidate set \( C_k \) contains nodes extended from \( \Lambda_{k-1} \). Suppose that \( \Lambda_{k-1} \subset \Lambda \), i.e. the algorithm have selected correctly nodes in \( \Lambda \). If \( r_{k-1} \neq 0 \), there is at least one node in \( C_k \) that is in \( \Lambda \), or \( \Lambda \cap C_k \neq \emptyset \), since all non-zero coefficients in \( x \) are connected in a rooted tree.

Note that:
\[
(\Lambda \setminus \tilde{C}_k) \cap \Lambda_{k-1} = (\Lambda \setminus (\Lambda_{k-1} \cup C_k)) \cap \Lambda_{k-1} = \emptyset
\]
and

$$(\Lambda \cap C_k) \cap \Lambda_{k-1} = \Lambda_{k-1} \cap C_k = \emptyset,$$

since $\Lambda_{k-1} \subset \Lambda$ and $\Lambda_{k-1} \cap C_k = \emptyset$.

From (2.17), $|c_j^{(k-1)}| = |x_j|$ for $j \in \Lambda \setminus \tilde{C}_k$ and for $j \in \Lambda \cap C_k$, since $\Lambda_{k-1} \cap C_k = \emptyset$ and $\Lambda_{k-1} \subset \tilde{C}_k$.

Let $n_0$ be the minimum level of nodes in $\Lambda \setminus \tilde{C}_k$, then

$$\Lambda \setminus \tilde{C}_k = \Lambda \cap \left\{ \bigcup_{n \geq n_0} L_n \right\} = \bigcup_{n \geq n_0} \{ \Lambda \cap L_n \}. \quad (2.21)$$

First, consider the left hand side of (2.19)

$$\max_{j \in \Lambda \setminus \tilde{C}_k} |c_j^{(k-1)}| = \max_{j \in \Lambda \setminus \tilde{C}_k} |x_j| = \max_{j \in \cup_{n \geq n_0} \{ \Lambda \cap L_n \}} |x_j| = \max_{j \in \cup_{n \geq n_0} L_n} |x_j| = \max_{j \in L_{n_0}} |x_j|,$$

where the decaying of coefficients across levels has been taken into account.

Next, consider the right hand side of (2.19)

$$\max_{j \in \Lambda \cap \tilde{C}_k} |c_j^{(k-1)}| \geq \max_{j \in \Lambda \cap C_k} |c_j^{(k-1)}| = \max_{j \in \Lambda \cap C_k} |x_j| \geq \max_{j \in L_{n_0} - 1} |x_j| \geq \max_{j \in L_{n_0}} |x_j|. \quad \Box$$

**Theorem 6.** Let $\mathbf{x}$ be a vector conforming to the sparse-tree model with $K$ non-zero elements. TOMP will recover $\mathbf{x}$ correctly if the sensing matrix $\mathbf{A}$ satisfies

$$\alpha \mu_1(K) + \mu_1(K - 1) < 1, \quad (2.22)$$

where $\alpha \in [0, 1]$ is the relaxation parameter and $\Omega$ is the index set of all columns of matrix $\mathbf{A}$.

**Proof.** Suppose that $\mathbf{x}$ conform to the sparse-tree model and $\mathbf{x}$ is exactly
sparse with \( K \) non-zero elements. Consider the case \( \alpha = 1 \), in which the finalist set \( F_k \) contains only one index.

At the \( k^{th} \) iteration, the algorithm will find the correct indices in \( \Lambda \) if

\[
\max_{j \in \mathcal{C}_k \cap \Lambda} |\langle r_{k-1}, a_j \rangle| > \max_{j \in \mathcal{C}_k \setminus \Lambda} |\langle r_{k-1}, a_j \rangle|.
\]

(2.23)

Since we have \( |\langle r_{k-1}, a_i \rangle| = 0 \quad \forall i \in \Lambda_{k-1} \), condition (2.23) is the same as

\[
\max_{j \in \mathcal{C}_k \cap \Lambda} |\langle r_{k-1}, a_j \rangle| > \max_{j \in \mathcal{C}_k \setminus \Lambda} |\langle r_{k-1}, a_j \rangle|,
\]

(2.24)

where \( \mathcal{C}_k \) is expanded to \( \tilde{\mathcal{C}}_k = \mathcal{C}_k \cup \Lambda_{k-1} \).

Consider the right hand side of (2.24):

\[
\text{RHS} = \max_{j \in \mathcal{C}_k \setminus \Lambda} \left| \sum_{i \in \Lambda} c_i^{(k-1)} \langle a_i, a_j \rangle \right|
\leq \max_{i \in \Lambda} |c_i^{(k-1)}| \max_{j \in \mathcal{C}_k \setminus \Lambda} \left| \sum_{i \in \Lambda} \langle a_i, a_j \rangle \right|
\leq \max_{i \in \Lambda} |c_i^{(k-1)}| \mu_1(\Omega)(K),
\]

where \( \Omega \) denotes the set of all indices of coefficients of \( x \).

Next, consider the left hand side of (2.23):

\[
\text{LHS} = \max_{j \in \mathcal{C}_k \cap \Lambda} \left| \sum_{i \in \Lambda} c_i^{(k-1)} \langle a_i, a_j \rangle \right|
= \max_{j \in \mathcal{C}_k \cap \Lambda} \left| c_j^{(k-1)} + \sum_{i \in \Lambda \setminus \{j\}} c_i^{(k-1)} \langle a_i, a_j \rangle \right|
\geq \max_{j \in \mathcal{C}_k \cap \Lambda} \left| c_j^{(k-1)} \right| - \sum_{i \in \Lambda \setminus \{j\}} \left| c_i^{(k-1)} \right| \max_{j \in \mathcal{C}_k \cap \Lambda} \sum_{i \in \Lambda \setminus \{j\}} \left| \langle a_i, a_j \rangle \right|
\geq \max_{j \in \mathcal{C}_k \cap \Lambda} \left| c_j^{(k-1)} \right| - \max_{i \in \Lambda} \left| c_i^{(k-1)} \right| \max_{j \in \mathcal{C}_k \cap \Lambda} \sum_{i \in \Lambda \setminus \{j\}} \left| \langle a_i, a_j \rangle \right|
\geq \max_{j \in \mathcal{C}_k \cap \Lambda} \left| c_j^{(k-1)} \right| - \max_{i \in \Lambda} \left| c_i^{(k-1)} \right| \mu_1(\Omega)(K - 1)
= \max_{j \in \Lambda} |c_j^{(k-1)}| - \max_{j \in \Lambda} |c_j^{(k-1)}| \mu_1(\Omega)(K - 1).
\]

The last step follows from Lemma 5. Thus, the condition such that the
algorithm determines the column(s) correctly at $k^{th}$ iteration is:

$$\mu_1(K) + \mu_1(K - 1) < 1. \quad (2.25)$$

In the general case, when $\alpha < 1$, there will be more than one node in the finalist set $F_k$. Following the same steps as in the above proof, the condition such that after (2.8) step, there is at least one correct node in $F_k$, is

$$\alpha \mu_1(K) + \mu_1(K - 1) < 1. \quad (2.26)$$

Since the correct nodes must minimize the residual, after the final selection step (2.9), $i_k$ must be in $\Lambda$. Given that the signal follows the sparse-tree model, the history set $H(i_k)$ contains indices of the optimal nodes. Consequently, TOMP will correctly recover the signal.

2.6.2 Effect of Parameter Variations

Since TOMP selects entries by expanding a selection tree, the final selected set is a connected sparse-tree. Moreover, only tree branches that lead to the smallest residual via orthogonal projection are selected at each iteration.

The parameters $d \in \mathbb{Z}$, $d \geq 1$ and $\alpha \in [0, 1]$ are the tuning parameters for TOMP. Larger $d$ leads to larger candidate sets, which allows us to reach further down significant coefficients of $x$, but at the cost of more computation per iteration. On the other hand, small $d$ helps to reduced computational cost at each iteration by limiting the number of candidates being considered but the finest levels of the tree might not be reached. If the signal strictly conforms to the proposed sparse-tree model, especially property 4 in Definition [1], the finest level coefficients will have small enough magnitudes to have significant effect on the reconstruction quality. Otherwise, the reconstruction quality will be affected if $d$ is too small. Thus, by varying $d$, one can control the trade-off between computational cost and robustness of the algorithm.

The relaxation parameter $\alpha$ allows further restriction of the search space to the finalist set by a fast evaluation of the inner products in (2.8) instead of a costly evaluation of the residual norms in (2.9). Smaller values of $\alpha$ lead to bigger finalist sets, which means more accurate selection, but also at the
cost of increased computation.

Some special cases for $d$ are:

- $d = \infty$ means the search space contains every node.

- $d = 1$ means only one new node, which is directly connected to the already selected set, is selected at each iteration. In this case the selection step (2.9) of TOMP can be achieved via evaluating inner products with residual $r_{k-1}$.

Similarly, the special cases of $\alpha$ are:

- $\alpha = 0$ leads to an exhaustive search of all possible history sets within the candidate set to determine the one leading to smallest residual.

- In general $\alpha = 1$ means only one finalist is selected at each iteration. In this case TOMP is almost like OMP except that TOMP selects a whole set $H(i_k)$ rather than only a single $i_k$. This selection approach ensures that the recovered signal will have the tree structure. If the signal satisfies our assumption $P_2$, this modification leads to the correct reconstruction, since coefficients in $H(i)$ are significant whenever coefficient $i$ is significant.

### 2.6.3 Computational Complexity Analysis

In this section, the computational complexity of TOMP, in terms of the number of multiplications, will be analyzed and compared with several algorithms, including OMP [18, 6], ModelCS (model-based CoSaMP) [7] and TSWCS (Tree-Structure Wavelet Compressed Sensing) [12]. Since the number of iterations are different for each method, the average computational cost per iteration will be computed and compared. The results are summarized in Table 2.1.

The comparison that we provide in this section is for the recovery of a signal that has the sparse-tree structure in the wavelet domain. Let $N$ be the length of the signal, $K$ be the number of non-zero coefficients and $M$ be the number of measurements. Since the mappings between the nodes on the tree and the indices are one-to-one, the terms nodes and indices are used interchangeably in the following analysis.
Complexity Analysis for TOMP

Assume that the signal $x$ to be recovered is an $L$-level wavelet decomposition of a length-$N$ signal $s$. Let $N_0$ be the number of scaling coefficients of $x$. Then $N_0 = \frac{N}{2^L}$.

1. Initialization step

In the initialization phase, all $N_0$ columns of $A$ which correspond to the scaling coefficients are selected. Let $\Lambda_0$ be the initial set of selected indices. The measurement vector is projected onto the space spanned by these columns. In the Gram-Schmidt implementation, these columns are orthonormalized with each other, which has the complexity of $O(MN_0^2)$ multiplications. Consequently, the complexity for the initialization step is $O(MN_0^2)$.

2. The first iteration

In the first iteration of the algorithm, the candidate set $C_1$ contains the indices of the nodes in the first $d$-level of the wavelet trees. For $N_0$ binary wavelet trees, $C_1$ has $N_0(2^d - 1)$ candidates.

To find the finalist set $F_1$, $N_0(2^d - 1)$ inner products between the residual and the columns of $A$ with indices in $C_1$ must be computed. This step has the complexity of $O\left(MN_0(2^d - 1)\right) \approx O\left(MN_0(2^d)\right)$ multiplications.

For each finalist in $F_1$, the algorithm estimates the residual that would be formed if the node and its ancestors are selected. To do that, the corresponding columns of $A$ are orthonormalized against the previously selected columns with indices in $\Lambda_0$ and against each other. In TOMP, the candidate set is formed by extending downward $d$ levels from the selected nodes. Thus, each finalist has at most $d - 1$ ancestors that have not been selected from the previous iterations. The maximum complexity for this orthonormalization step is $2M \sum_{j=0}^{d-1} N_0 + j \approx O\left(MN_0d + Md^2\right)$.

The total complexity for the first iteration is thus

\[ O\left(MN_02^d\right) + O\left(|F_0| \left(MN_0d + Md^2\right)\right), \]

where $|F_0|$ denotes the cardinality of the finalist set $F_0$. Typically, after
the thresholding step in (2.8), there are only several finalists in the set \( F_0 \). Consequently, the complexity of the first iteration is \( \mathcal{O}(M N_0 2^d) \).

3. From second iteration onward

From the second iteration onward, at each iteration, at most \( d \) nodes from the candidate set are selected and their descendants are added to the candidate set. Thus, the size of the candidate set increases by a small constant at each iteration. The complexity at each following iterations can be approximated by that of the first iteration, which is \( \mathcal{O}(M N_0 2^d) \).

With this estimation, the complexity of \( k \) iterations of the algorithm would be \( \mathcal{O}(k 2^d M N_0) \). The total complexity for TOMP is \( \mathcal{O}(MN_0^2) + \mathcal{O}(k M N_0 2^d) \).

Since TOMP selects many indices at each iteration, the number of iterations \( k \) for TOMP is less than the number of non-zero coefficients \( K \) in signal \( x \). Moreover, for most practical applications of the tree model, the number of roots \( N_0 \) is small compared to the number of nodes on the tree. As a result, the total complexity of TOMP is bounded by \( \mathcal{O}(K M N_0 2^d) \). Finally, the average complexity per iteration for TOMP is approximately \( \mathcal{O}(M N_0 2^d) \approx \mathcal{O}(\frac{MN}{2^{d-1}}) \).

Complexity Analysis for OMP, ModelCS and TSWCS

In OMP, at the \( n^{th} \) iteration, \( N - n + 1 \) inner products must be computed between \( r_{n-1} \) and the remaining columns. After that, the selected column is orthonormalized against all \( n - 1 \) columns that have been selected from the previous iterations. OMP stops after \( K \) iterations and the total complexity is \( \mathcal{O}(K M N + MK^2/2) \) multiplications. Thus, the average complexity per iteration for OMP is \( \mathcal{O}(M N + MK/2) \) multiplications.

CoSaMP improves over OMP by selecting many columns at each iteration and refining them later, which requires a smaller number of iterations. ModelCS is developed from CoSaMP by applying a tree approximation algorithm during the pruning step to maintain the tree structure on the selected columns. As a result, ModelCS has a complexity of OMP plus the complexity of the tree-approximation step. Let \( C \) be the complexity of the tree-approximation algorithm, the total complexity of ModelCS for \( k \) iterations
Table 2.1: Average computational complexity per iteration of the algorithms, in terms of number of multiplications, to recover a length-$N$, $K$-sparse signal with sparse-tree structure in the wavelet domain from $M$ random measurements. In ModelCS, $C$ is the cost of the tree-approximation algorithm being used. In TOMP, $L$ is the number of levels and $d$ is the downward extending parameter.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Average number of multiplications per iteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMP</td>
<td>$\mathcal{O}(MN + MK/2)$</td>
</tr>
<tr>
<td>ModelCS</td>
<td>$\mathcal{O}(MN + C)$</td>
</tr>
<tr>
<td>TSWCS</td>
<td>$\mathcal{O}(MN^2)$</td>
</tr>
<tr>
<td>TOMP</td>
<td>$\mathcal{O}(\frac{MN}{2^{L-d}})$</td>
</tr>
</tbody>
</table>

is $\mathcal{O}(kMN + Mk^2/2 + kC)$ multiplications. The complexity per iteration is $\mathcal{O}(MN + Mk + C) \approx \mathcal{O}(MN + C)$ multiplications.

TSWCS uses Gibbs sampling and Markov Chain Monte Carlo to infer the posterior distribution of each element in the unknown signal. At each iteration, TSWCS needs to sample each element of the signal sequentially, and thus the total cost of TSWCS is significantly higher than OMP. The computational complexity of TSWCS is approximately $\mathcal{O}(kMN^2)$ multiplications, where $k$ is the number of iterations. The average complexity per iteration for TSWCS is $\mathcal{O}(MN^2)$ multiplications.

### 2.7 Experimental Results

In this section, three main experiments are presented. First, to demonstrate the correctness of the recovery of tree structure, a test signal is recovered using different methods: OMP [18, 6], CoSaMP [19], ModelCS (model-based CoSaMP) [7], TSWCS (Tree-Structure Wavelet Compressed Sensing) [12], and our proposed TOMP. Second, the average reconstruction signal-to-noise ratios (SNRs) of different methods are compared at different numbers of measurements on a random piecewise smooth signal. The execution times are also compared. Finally, the *phase diagram* [21] of our proposed method is generated and presented to show the phase transition of the algorithm under different regimes.

We obtained the TSWCS and ModelCS code from the corresponding authors. For a fair comparison between different methods, we modify the Mod-
elCS code to use MATLAB Wavelet Toolbox instead of Rice Wavelet Toolbox as in the original code. This modification allows us to run all algorithms on the same input. Moreover, ModelCS requires an estimation of the signal sparsity as input. In our experiments, we run ModelCS multiple times with varying values and report the result which has maximum SNR. We vary the sparsity from 10 percent the length of the signal to 50 percent the number of measurements.

The reconstruction quality is measured using reconstruction signal-to-noise ratio (SNR), defined as:

\[
\text{SNR} = 10 \log_{10} \left( \frac{\sigma_x^2}{\text{MSE}(x, \hat{x})} \right) \text{dB},
\]

where \( \sigma_x^2 \) is the variance of the elements of true vector \( x \) and MSE(\( x, \hat{x} \)) is the mean squared error between the true vector \( x \) and the reconstruction \( \hat{x} \).

2.7.1 Reconstruction Examples

In this experiment, different methods are used to reconstruct a perfect sparse-tree signal from a small number of measurements. The test signal is a piecewise polynomial with maximum order of three and one discontinuity. The signal is decomposed by a 4-level wavelet transform using Daubechies wavelets of 4 vanishing moments. The measurement matrix is a random matrix of i.i.d. Gaussian entries with normalized columns. In this setup, the length of the input signal is chosen to be 64 and the number of measurements is 35. For TSWCS and ModelCS, the recommended values of the parameters are used. For our proposed TOMP, \( dlev = 2 \) and \( \alpha = 0.9 \). Figure 2.2 shows the reconstructed signals. Figure 2.3 displays the reconstructed trees, where black nodes are non-zero nodes. As shown in this figure, TOMP, TSWCS and ModelCS maintain the tree structure in the recovered signal very well, whereas OMP and CoSaMP fails to capture it.

2.7.2 Performance Comparison

In this experiment, a test signal is reconstructed from different numbers of measurements. For each number of measurements, 100 tests are run with ran-
Figure 2.2: An example piecewise polynomial signal of length 64 and its reconstructions from 35 linear measurements using TOMP, TSWCS, ModelCS, OMP and CoSaMP.
Figure 2.3: The wavelet trees of the original signal and reconstructed signals using TOMP, TSWCS, ModelCS, OMP and CoSaMP. Black nodes are non-zero nodes.
domly generated measurement matrices, and then the average reconstruction SNRs are computed. The input signal is a random piecewise smooth signal with 12 discontinuities. The signal has a length of 1024 samples. The wavelet coefficients are computed by a 6-level wavelet decomposition using Daubechies wavelets with 4 vanishing moments. This signal is not exactly sparse and the coefficients are connected loosely to a tree structure. The values of the parameters of TOMP are $\alpha = 0.9$ and $dlev = 2$.

Figure 2.4 compares the average performance of different algorithms in recovering a random piecewise smooth signal of length 1024, in terms of reconstruction quality. Although TSWCS obtains very high SNRs, it is more computationally expensive as opposed to our algorithm, as will be discussed in Section 2.8.

![Figure 2.4: Performance comparison between different algorithms for the reconstruction of a random piecewise smooth signal of length 1024 from different numbers of measurements](image)

2.7.3 Phase Diagram

*Phase diagrams* [21] are used to describe visually the performance of a reconstruction algorithm. A phase diagram shows the probability of successful recovery of a sparse signal under different regimes or problem suites. A
problem suite \( S(K, M, N) \) is defined as a collection of a set of random matrices of size \( M \times N \) and a set of \( K \)-sparse vectors. For recovery algorithms based on \( l_1 \) relaxation techniques, it has been observed that the transition from success to failure occurs sharply along a predictable line in the \( \rho - \delta \) plane, where \( \rho = K/M \) and \( \delta = M/N \) for problem suite \( S(K, M, N) \). This is referred to as the phase transition phenomenon. The empirically observed locations of the phase transition can be predicted accurately in theory for \( l_1 \)-based methods. However, for greedy methods, there has been no theory for the prediction of the phase transition line, although the phenomenon can be still observed empirically in many cases.

In this experiment, the phase diagram for OMP and TOMP are computed and compared for sparse-tree signals. For each set \( (K, M, N) \), a random sparse-tree signal with sparsity of \( K \) is generated, together with a random measurement matrix of dimensions \( M \times N \). Figure 2.5 shows the phase diagram comparison between TOMP and OMP. The figure displays the number of coefficients that are different from the original coefficients, with darker shade represents higher number of incorrect coefficients. Both algorithms exhibit the phase transition phenomenon, with sharper transition at higher number of measurements. From the two diagrams, we can see that at the same number of measurements, TOMP can recover signals that have a higher ratio \( K/M \) at higher success rates.

2.8 Discussion

The above experiments show the competitive performance of TOMP for 1-dimensional sparse-tree signals. Compared with OMP, TOMP and other tree-based methods give higher reconstruction quality, especially at a low number of measurements. The TSWCS method gives the highest SNR in most cases. At a low level of measurement noise, TOMP consistently gives reconstruction quality comparable to TSWCS while being much faster. A disadvantage of TSWCS is its higher computational complexity, since at each iteration, the algorithm needs to sample each element of the signal to be recovered using Gibbs sampling. In contrast, at each iteration, TOMP searches for the location of only significant elements. Compared with ModelCS, TOMP also has less computational complexity since at each iteration...
Figure 2.5: Phase diagram of TOMP in comparison with OMP. The brighter shade shows higher success rate (i.e., fewer number of coefficients that are different from the original signal)
of ModelCS, the algorithm has to compute the inner product of the residual with all columns of the measurement matrix to “sense” the location of significant coefficients. In contrast, TOMP only needs to compute the inner product between the residual and the candidate columns. The highest computational cost of TOMP is the projection onto the history nodes for each finalist, which could be reduced by increasing the relaxation parameter $\alpha$ (which in effect reduces the number of finalist sets).

As discussed in Section 2.6.2, there is a trade-off between computational cost and robustness of the algorithm to signals that do not conform to the sparse-tree model. This trade-off is controlled by parameter $d$. If $d$ is small then the computational cost at each iteration will be low but the finest levels of the tree might not be investigated. On the other hand, a large value of $d$ ensures that the algorithm will find significant coefficients at deeper levels but the computational cost will be higher. For signals that strictly conform to the sparse-tree model, the finest level coefficients will have small magnitudes and do not contribute much to the reconstructed signal. Thus, $d$ can be set to a small value for these signals to reduce the computational cost.

Although TOMP is specifically designed for sparse-tree signals, the algorithm still works with small deviations from the model. This is empirically proved by the results with piecewise smooth signal in Figure 2.4.

For natural images, a separable wavelet transform does not effectively exploit the fact that discontinuities are formed along geometrically smooth curves. With a typical image of size $256 \times 256$, a 2-dimensional wavelet transform gives shallow trees with a maximum height of 8. TOMP does not work effectively in this situation. The recovery of images will be investigated in an upcoming paper with a more effective geometric 2-dimensional decomposition, such as curvelets \cite{35} and contourlets \cite{36}, where significant coefficients are successively localized in a tree structure in both location and direction.

2.9 Concluding Remarks

Most existing compressed sensing recovery algorithms still only exploit the sparse prior of signals, regardless of any signal structure that may present. Despite an increasing interest in model-based compressed sensing in the past
few years, the application of the proposed methods has still been limited in practice. In this chapter, we present a simple yet effective algorithm that exploits the sparse-tree structure of the signal for signal reconstruction. The sparse-tree inverse problem has been formulated and its usefulness has been justified. Based on that, the Tree-based Orthogonal Matching Pursuit (TOMP) algorithm is proposed and analyzed in detail, both theoretically and empirically. The experimental results confirm the state-of-the-art performance of TOMP with significantly lower computational cost.
Chapter 3

A New Method for Registration of 3D Point Clouds

3.1 Introduction

In 3D imaging, people try to capture and reconstruct 3D models of objects or scenes. Due to the limited field of view of the capturing devices and occlusion, multiple views of an object are captured and then merged together to generate the complete 3D model of it. Registration is the process of aligning different views correctly before merging. Thus, registration plays an important role in 3D imaging applications.

Many methods have been proposed for registering 3D point clouds and models. The Iterative Closest Point algorithm (ICP) [8, 9, 10] is perhaps the most popular method to align two point clouds, given a rough estimation of the transformation between them. Many variants have been proposed to improve the performance of ICP [11]. These methods work well on clean point clouds, such as those given by a laser scanner, and do not assume any structure in the point sets.

Recently, commodity depth cameras have been gaining more interest due to their low cost and fast capture rate. Many methods have been proposed to use these cameras as a cheaper and faster way to capture 3D data. The main challenges with current depth cameras are the low resolution or quality of the acquired depth images. For example, depth data from time-of-flight (ToF) cameras are affected by both random noise and bias noise [38, 39] whereas depth data from structured-light (SL) cameras usually have missing pixels because of occlusion. Due to these problems, ICP does not perform as well on them. Several approaches have been proposed to improve the robustness of ICP on noisy data, for example, by adding heuristic pruning and weighting of correspondences.

In this work, we propose a new method, named the Signed Distance Reg-
istration (SDR), to register noisy point clouds based on the Implicit Moving Least Squares (IMLS) function. We assume the point clouds are sampled from the same underlying surface under some rigid transformation. We also provide a detailed analysis of the method and compare it with related methods. Moreover, we demonstrate the applicability of our method in 3D reconstruction with a complete pipeline to reconstruct a 3D model from a sequence of depth images.

The main contributions of this work are:

- Propose the Signed Distance Registration (SDR) algorithm, a novel registration method based on the IMLS function.

- Integrate the proposed method in a complete 3D reconstruction process.

This chapter is organized as follows. First, relevant research will be reviewed in Section 3.3. After that, some background material will be presented in Section 3.4 to provide the notations for later sections. Our proposed method will be introduced in Section 3.5. Section 3.6 will provide a theoretical analysis and comparison of different approaches. Experimental results will be provided in Section 3.8. The paper will be concluded with Section 3.9.

3.2 Problem Statement

Given two point sets, the source set $P = \{p_i\}_{i=1,...,N_p}$ and the destination set $Q = \{q_j\}_{j=1,...,N_q}$, which are sampled on the same underlying surface under slightly different viewpoints, assume that the two point sets are related by a rigid transformation, which includes a rotation and a translation, and are roughly registered (i.e. the transformation is small between them).

The registration problem is to find a rotation matrix $R$ and a translation vector $t$ that align the source set $P$ to the destination set $Q$. 
3.3 Related Work

3.3.1 Registration

Registration is an important step in 3D reconstruction that determines the quality of the final result. It is required to correctly align depth images from different viewpoints together, before fusing them into a common 3D model. There are many methods for registration, which can be grouped into two main categories: coarse registration and fine registration. Coarse registration methods provide a rough estimate of the relative transformation between two point sets, which will be refined with a fine registration method. In this work, we only focus on the fine registration step.

ICP is the most popular method for fine registration. Its name was proposed in [8] but the method was also independently introduced in [9, 10]. The main idea of ICP is iterating between matching correspondences and computing transformation. In [8], ICP was proposed to register an unorganized point cloud to a general geometric entity and the correspondences are selected based on closest Euclidean distance. Chen and Medioni [9] assumed that a surface normal is included with each point, which is used in the correspondence matching step. Zhang [10] introduced the weighting and rejection steps to deal with outliers and wrong correspondences. A survey of different variants of ICP is presented in [11], with the introduction of a fast variant of ICP. Fitzgibbon [40] presented the LM-ICP method, which directly minimizes the cost function using Levenberg-Marquadt algorithm and applied robust kernels to improve the robustness.

There is another class of registration methods that avoid the correspondence matching problem. In [41], the author proposed a method for registration and integration of range images by matching the signed distance fields. The signed distance fields are estimated from two point set and the difference between them is minimized to compute the transformation parameters. [42] and [43] proposed a new registration method by minimizing a cost function defined on the squared distance function. These authors proposed to locally approximate the squared distance function with a quadratic function and showed that their method can achieve locally quadratic convergent rate.

Statistical methods are also proposed for the point cloud registration problem. In [44, 45], the authors represent the point sets using Gaussian mixture
models and align them by minimizing a statistical discrepancy. [46] proposed a particle filtering approach to register two point sets whereas [47] introduced an EM-like algorithm. Maier-Hein et al. developed the Anisotropic ICP (A-ICP) algorithm that takes into account anisotropic and inhomogeneous localization errors.

3.3.2 3D Reconstruction with Depth Camera

The depth camera enables us to obtain depth information at video rate with the cost of low resolution and quality. The depth camera has been successfully applied in applications that only require coarse depth information at high speed, such as real-time gesture recognition. Its low resolution, however, poses a challenge for 3D reconstruction problem.

In [39], the authors proposed an initial attempt to reconstruct a 3D model with a ToF depth camera. In this approach, a 2D depth image super-resolution technique, LidarBoost [48], is applied on neighboring depth maps to create a sequence of high-resolution depth maps. After that, a semi-rigid global registration method is used to register these super-resolved depth maps together and reconstruct the 3D model.

The first method that is able to achieve real-time reconstruction with depth camera is KinectFusion [49, 50, 51]. In this approach, the authors proposed to register and fuse subsequent depth maps on to a global 3D volumetric model. The registration step is performed by a coarse-to-fine ICP algorithm which uses projective association for finding correspondences and minimizes a point-to-plane error metric. The model is represented by a truncated signed distance function (TSDF) [52]. Each new frame is registered to the frame synthesized at previous viewpoint by ray casting the 3D model. The whole pipeline is implemented on GPU to perform in real time. A similar method was independently proposed in [53] in which the model is represented by IMLS function [54] and new depth frame is registered directly onto the model.

Recently, [55] proposed another method for real-time camera tracking and 3D reconstruction with depth camera. In the method, the authors improved KinectFusion by skipping the ray casting step and registering a new frame directly onto the model, instead of registering them to a synthetic frame. The authors also proposed to parameterize the transformation with Lie al-

37
gebra [56], a common practice in robotics. The registration is performed by minimizing an objective function computed from the current model.

3.4 Background

3.4.1 Depth Capturing

There are many ways to capture depth information which can be classified into two main categories: passive and active techniques. Passive techniques include stereo vision and structure-from-motion (SfM) where depth information is computed from passive images of the scene. Active techniques include laser scanner, structured-light (SL) and time-of-flight (ToF) cameras, which actively illuminate the scene with a laser or an IR source.

Stereo vision and SfM derive depth information by triangulating corresponding points between multiple 2D images. These methods do not work well on textureless scenes or scenes with repetitive textures. In a textureless scene, there are not enough correspondences to reliably infer depth information. On the other hand, a scene with many repetitive textures leads to many wrong correspondences due to ambiguity in matching features across views.

High-quality point clouds can be captured with a laser scanner, which projects a laser beam to the surface, captures the reflected beam and triangulates to get the position of the surface point. The laser beam is scanned over the object surface to obtain a set of points sampled on the surface. This method provides high-quality data at the cost of long capture time and expensive device.

In [57], the authors introduced a faster way to capture 3D information by using a projector and a camera. The projector projects a known pattern onto the object, which is captured by the camera. The captured image and the known pattern are used to infer depth information. This system is an early version of SL camera, which is the underlying working principle of the Kinect camera.

The Kinect depth camera integrates an IR light source and an IR camera to create a portable depth camera, together with a normal color camera. Given the fast capture rate of Kinect, which is up to 30 frames per second, it has been employed in several applications which requires real-time response such
as gesture recognition for gaming and user-machine interaction. Recently, in [50, 51], the authors proposed a method for real-time 3D reconstruction using Kinect camera, by combining registration and integration together. A method for simultaneously localization and mapping (SLAM) using Kinect was proposed in [49].

There is another type of depth camera that is based on time-of-flight technology. With these cameras, a modulated IR light is projected on to the scene and the depth is inferred from the phase correlation between the projected and the reflected light. ToF depth maps have a higher level of noise than SL depth map and is affected by a systematic bias [38].

3.4.2 Signed Distance Function and IMLS Method

Surface representation is an important factor in 3D reconstruction. In this work, we are interested in an implicit representation of surfaces with the signed distance function. A signed distance function is a scalar function $f(x)$ that provides the signed distance from a location $x \in \mathbb{R}^3$ to the surface. The value of $f$ is positive for points outside, negative for points inside and zero for points on the surface.

Given the signed distance function $f$, the surface $S$ represented by $f$ can be reconstructed as:

$$S = \{ x \in \mathbb{R}^3 | f(x) = 0 \}. \hspace{1cm} (3.1)$$

This reconstruction can be performed with Marching Cubes [58] or Ray Tracing [59] algorithms.

In practice, the surface is sampled by a set of points and it is necessary to estimate the signed distance function of the underlying surface. Implicit Moving Least Squares (IMLS) [54, 60] is one simple method for this purpose.

Given a set of $N$ points $\{p_i\}_{i=1,...,N}$ and the corresponding surface normals at those points $\{n_i\}_{i=1,...,N}$, IMLS method computes the function $f_{\text{IMLS}}$ as

$$f_{\text{IMLS}}(x) = \frac{\sum_{i=1}^{N} n_i^T (x - p_i) \phi_i(x)}{\sum_{i=1}^{N} \phi_i(x)}, \hspace{1cm} (3.2)$$

where

$$\phi_i(x) = \exp \left( -\frac{\|x - p_i\|^2}{2\sigma^2} \right). \hspace{1cm} (3.3)$$
and $\sigma$ is the smoothing parameter. Intuitively, the method locally approximates a signed distance function at each point by fitting a micro-plane and blends these local approximates together by a weighted average.

The gradient of the function $f_{\text{IMLS}}$ can be derived from Equation 3.2 as

$$\nabla_x f_{\text{IMLS}}(x) = \frac{\sum_{i=1}^{N} n_i^T (x - p_i) \nabla_x \phi_i(x) + \sum_{i=1}^{N} \phi_i(x) n_i - \sum_{i=1}^{N} f(x) \nabla_x \phi_i(x)}{\sum_{i=1}^{N} \phi_i(x)}.\tag{3.4}$$

### 3.4.3 Iterative Closest Point (ICP) Algorithm

Iterative Closest Point (ICP) [8] is the most popular method for registration of 3D point set. This method iteratively matches points from one point set with those in the other set and computes the transformation to bring them together.

Let $P$ is the source set and $Q$ is the destination set. ICP iterates between two following steps:

1. **Finding correspondences** For each point in the source set, a corresponding point in the destination set is found. There are several heuristics proposed for finding correspondence. The original way is choosing the point in the destination set that is closest in Euclidean distance to the source point.

   $$j(i) = \arg\min_{j=1,2,\ldots,N_q} \| q_j - p_i \|_2^2.\tag{3.5}$$

   The problem with this method is the high computational complexity. A straightforward implementation will have a cost of $O(N_P N_Q)$ for computing correspondences. This can be speeded up by using k-d tree [61] but there is still cost to build the tree.

   A faster way is the projective association method [62]. In this method, each point from the source set is projected onto the image plane of a camera that is placed at the viewpoint of the destination frame. The correspondence is selected as the point that is projected on the corresponding pixel in the image. The advantage of this method is the $O(N_P)$ computational cost. Figure 3.1 compares these two correspondence matching method.
Figure 3.1: The difference between closest point and projective association method for finding correspondences. The former selects the point on the destination set that is closest to the point on the source set in Euclidean distance. The latter projects the source point onto the destination image plane and chooses the destination point which is projected to the same location.

2. Finding the transformation Given the correspondences, the rotation $R$ and translation $t$ to align the point sets can be found by minimizing the following cost function:

$$J_{\text{point-point}} = \sum_{i=1}^{N_p} \|Rp_i + t - q_{j(i)}\|_2^2,$$  \hspace{1cm} (3.6)

where $q_{j(i)}$ is the correspondence of $p_i$ in $Q$. This cost function is based on point-to-point error metric and it has closed form solutions [63, 64, 65].

If the point sets include surface normal at each point, another cost function based on point-to-plane error metric can be used [9]. The difference between these two cost function is shown in Figure 3.2. Point-to-plane error metric has been shown to improve convergence rate and robustness [11]. Letting $n_i^j$ be the surface normal at point $q_i$, the point-to-plane cost function is:

$$J_{\text{point-plane}} = \sum_{i=1}^{N_p} \left[ n_{j(i)}^T (Rp_i + t - q_{j(i)}) \right]^2.$$  \hspace{1cm} (3.7)

To minimize $J_{\text{point-plane}}$, $R$ can be linearized by assuming small rotation
angle between two sets. An linear system of equations is formed from (3.7) and solved for \( R \) and \( t \). Otherwise, one can also use an iterative optimization method.

![Figure 3.2: The difference between point-to-point error metric and point-to-plane error metric. Point-to-point metric estimates the distance from a point to a surface by its distance to the closest sample of the surface. Point-to-plane error metric approximates the distance to the surface by the distance to the surface tangent at the closest sample.](image)

### 3.5 Proposed Algorithm

#### 3.5.1 Registration Using Signed Distance Function

Consider the two point sets \( P \) and \( Q \). Suppose that they are point samples of two surfaces \( S_P \) and \( S_Q \), which are indeed the same surface related by an unknown rigid transformation. The main goal of ICP is to find the transformation that brings these two surfaces together. To achieve that goal, the distance between the transformed source surface and the destination surface must be minimized. ICP tries to minimize the sum of squared distances between each source point and the destination surface.

In closest point ICP, the distance is approximated by the smallest point to point distance. Point-to-plane ICP approximates the distance by the distance between source point and tangent plane at corresponding target point.

Assume that the signed distance function \( f \) representing the target surface is available. Then the generalized cost function for registration can be defined as

\[
J_{SD} = \sum_{i=1}^{N_p} (f(Rp_i + t))^2.
\] (3.8)
In [42], the authors approximate the squared signed distance function by a local quadratic approximant. In this work, we proposed to use IMLS method to estimate the signed distance function. Originally, IMLS was developed to compute a good surface representation given a polygon mesh [54]. Kolluri [66, 60] proved a guarantee on reconstruction quality of surface using IMLS under uniform sampling. The guarantee for adaptive sampling was proved by Dey and Sun in [67].

The advantage of using the signed distance cost function is that the correspondence matching step is eliminated. The registration problem becomes an optimization problem:

\[
\{ R, t \} = \arg \min_{R,t} \sum_{i=1}^{N_p} (f(Rp_i + t))^2. \tag{3.9}
\]

Consider a general parameterization of the transformation \( T \) and let \( s \) be the parameter vector, then Equation 3.9 can be rewritten as

\[
s = \arg \min_s \sum_{i=1}^{N_p} (f_i(s))^2 \tag{3.10}
\]

\[
= \arg \min_s \sum_{i=1}^{N_p} (f_i(s))^2, \tag{3.11}
\]

where \( f_i(s) = f(T_s(p_i)) \).

### 3.5.2 Optimizing the Cost Function

Given the form of the cost function, the Gauss-Newton method can be used to solve the optimization problem. The advantage of this method is we do not need to compute the second derivative of \( f \).

Consider the \( k^{th} \) iteration of the algorithm. Setting the gradient of \( J_{SD} \) with respect to vector \( s \) to zero, we obtain

\[
\sum_{i=1}^{N_p} f_i(s) \nabla_s f_i(s) = 0. \tag{3.12}
\]
Approximate $f_i(s)$ by its first-order Taylor series, we have

$$f_i(s) \approx f_i(s^{(k-1)}) + \nabla_s^T f_i(s^{(k-1)}) (s - s^{(k-1)}), \quad (3.13)$$

where $s^{(k-1)}$ is the parameter vector obtained from the $(k-1)^{th}$ iteration.

Equation (3.12) becomes

$$\sum_{i=1}^{N_p} [f_i(s^{(k-1)}) + \nabla_s^T f_i(s^{(k-1)}) (s - s^{(k-1)})] \nabla_s f_i(s^{(k-1)}) = 0, \quad (3.14)$$

where we approximate $\nabla_s f_i(s) \approx \nabla_s f_i(s^{(k-1)})$.

The value of $s^{(k)}$ is computed as

$$s^{(k)} = s^{(k-1)} - A^{-1} b, \quad (3.15)$$

where

$$A = \sum_{i=1}^{N_p} \nabla_s f_i(s^{(k-1)}) \nabla_s^T f_i(s^{(k-1)}) \quad (3.16)$$

and

$$b = \sum_{i=1}^{N_p} f_i(s^{(k-1)}) \nabla_s f_i(s^{(k-1)}). \quad (3.17)$$

3.5.3 Algorithm

This section describes in detail our proposed Signed Distance Registration algorithm for registering the source point set $P$ to the destination point set $Q$.

**Algorithm 7** *(Signed Distance Registration algorithm).*

**INPUT**

- The source point cloud with normals $P = \{p_i, n_i^{(p)}\}_{i=1}^{N_p}$,
- The destination or target point cloud with normals $Q = \{q_j, n_j^{(q)}\}_{j=1}^{N_q}$,
- Smoothing parameter $\sigma$.

**OUTPUT**
• Parameter vector $s$ representing the rigid transformation to bring $P$ to $Q$.

PROCEDURE

1. Initialize vector $s$ as:
   \[
   s^{(0)} = s_0, \tag{3.18}
   \]
   where $s_0$ represents the identity transformation:
   \[
   T_{s_0}(x) = x. \tag{3.19}
   \]
   Set $k = 0$ and stop = FALSE.

2. While stop is FALSE:
   (a) $k = k + 1$.
   (b) For $i = 1, \ldots, N_p$, compute $f(T_{s^{(k-1)}(p_i)})$ and $\nabla_s s^{(k-1)} f(T_{s^{(k-1)}(p_i)})$, using Equations 3.2 and 3.4.
   (c) Compute $A$ and $b$ using Equations 3.16 - 3.17.
   (d) Compute update vector $\delta = A^{-1}b$.
   (e) Update parameter vector as:
   \[
   s^{(k)} = s^{(k-1)} - \delta. \tag{3.20}
   \]
   (f) If $||\delta||$ is less than a threshold, set stop = TRUE.

3. Return $s^{(k)}$.

3.5.4 A Variant of ICP with IMLS Matching

Another version of our proposed method can be considered as a variant of ICP. In this version, instead of optimizing a cost function, we follow the main steps of ICP and replace the correspondence matching step with another method based on signed distance function. In particular, consider a point $p_i$ in the source point cloud; we find the corresponding point $q_i$ of $p_i$ by
\[
q_i = p_i - f(p_i)\nabla_x (p_i), \tag{3.21}
\]
where \( f(p_i) \) and \( \nabla_x(p_i) \) are the value of the signed distance function and its gradient at \( p_i \). In this formula, \( q_i \) is found by projecting \( p_i \) on to the surface implied by the signed distance function \( f(x) \). To estimate \( f(x) \), we used the IMLS method as described above. Note that the correspondence of \( p_i \) is not necessarily a point in the original target set \( Q \). Thus, this method will work when the two point sets are sampled differently from the surface. We named this version IMLS-ICP. Figure 3.3 illustrates the matching step using IMLS-ICP.

![Figure 3.3: Point matching with the proposed IMLS-ICP algorithm. Source point is projected onto the surface implied by the signed distance function \( f \). This method allows registering points to a surface.](image)

**Algorithm 8 (IMLS-ICP registration algorithm).**

**INPUT**

- The source point cloud with normals \( P = \{ p_i, n_i^{(p)} \}_{i=1,...,N_p} \),
- The destination or target point cloud with normals \( Q = \{ q_j, n_j^{(q)} \}_{j=1,...,N_q} \),
- Smoothing parameter \( \sigma \).

**OUTPUT**

- Parameter vector \( s \) representing the rigid transformation to bring \( P \) to \( Q \).

**PROCEDURE**

1. Initialize vector \( s \) as:
   \[
   s^{(0)} = s_0, \quad (3.22)
   \]
where $s_0$ represents the identity transformation:

$$T_{s_0}(x) = x.$$  \hspace{1cm} (3.23)

Set $k = 0$ and stop = FALSE.

2. While stop is FALSE:

(a) $k = k + 1$.

(b) For $i = 1, \ldots, N_p$, compute $f(T_{s^{(k-1)}}(p_i))$ and $\nabla_{s^{(k-1)}f(T_{s^{(k-1)}}(p_i))}$, using Equations 3.2 and 3.4.

(c) Compute correspondence point $q_i$ using Equation 3.21.

(d) Estimate the normal vector $n_i^{(q)}$ using $\nabla_{s^{(k-1)}}f(T_{s^{(k-1)}}(q_i))$.

(e) Setting up the system of equations

$$n_i^{(q)^T}(R_{s^{(k-1)}}p_i + t_{s^{(k-1)}} - q_i) = 0, \quad i = 1, \ldots, N_p.$$  \hspace{1cm} (3.24)

(f) Solve [3.24] for parameter $s^{(k)}$.

(g) Update the source point set and normals

$$p_i = R_{s^{(k)}}p_i + t_{s^{(k)}},$$  \hspace{1cm} (3.25)

$$n_i^{(p)} = R_{s^{(k)}}n_i^{(p)}.$$  \hspace{1cm} (3.26)

(h) If the error between source set and target set is less than some threshold, set stop to TRUE.

3. Return $s^{(k)}$.

3.6 Analysis and Comparison

In this section, we provide a theoretical analysis of the proposed registration algorithm.

First, we look at the relationship between the algorithms and show that our proposed method approximates the surface by a smooth regression of the points whereas closest-point ICP with point-to-plane metric only performs a linear interpolation between the points.
Second, we analyze the behavior of the methods in flatland. We will look at a challenging 2D example where ICP fails and we demonstrate that by controlling the smoothing parameter $\sigma$, SDR can produce the correct optimal result after a small number of iterations.

3.6.1 Cost Function Analysis

For closest-point ICP, for each point in the source set, the closest point in the other set (based on Euclidean distance) is selected as the correspondence, following Equation 3.5.

When the error metric is point-to-plane distance, the error is defined for each pair of corresponding points as the distance from the source point to the tangent plane of the destination point at the correspondence. As explained in [11], this error metric allows two planes to slide over each other, thus enhancing the convergence rate. The point-to-plane cost function is defined in Equation 3.7.

By substituting the IMLS function definition in Equation 3.2 into the generalized cost function 3.8, we obtain

$$J_{IMLS} = \sum_{i=1}^{N_p} \left[ \frac{\sum_{j=1}^{N_q} (n_j^q)^T (R_{i} p_i + t - q_j) \phi_j(R_{i} p_i + t - q_j)}{\sum_{j=1}^{N_q} \phi_j(R_{i} p_i + t - q_j)} \right]^2. \quad (3.27)$$

Defining

$$\hat{w}_{i,j} = \frac{\phi_j(R_{i} p_i + t - q_j)}{\sum_{j=1}^{N_q} \phi_j(R_{i} p_i + t - q_j)}, \quad (3.28)$$

the cost function becomes

$$J_{IMLS} = \sum_{i=1}^{N_p} \left[ \sum_{j=1}^{N_q} \hat{w}_{i,j} (n_j^q)^T (R_{i} p_i + t - q_j) \right]^2. \quad (3.29)$$

With Equation 3.29, the distance from each source point to the destination surface is defined by a weighted average of its distances to the tangent planes at several destination points. The weight given to each destination point depends on its distance to the source point.

When the smoothing parameter $\sigma$ approaches zeros, $\hat{w}_{i,j}$ approaches a delta function and Equation 3.29 will become Equation 3.7 in the vicinity of the
surface. In this case, the proposed method reduces to closest-point ICP with point-to-plane error metric.

On the other hand, when the smoothing parameter \( \sigma \) is very large, \( \hat{w}_{i,j} \) approaches \( \frac{1}{N_q} \). In this case, the cost for each source point is the averaged point-to-plane distance from its to the tangent planes at all target points

\[
J_{IMLS}^{(\sigma \to \infty)} = \sum_{i=1}^{N_p} \left[ \frac{1}{N_q} \sum_{j=1}^{N_q} (n^t_j)^T (R_{p_i} + t - q_j) \right]^2. \tag{3.30}
\]

The different cost functions can be thought to imply different approximations of the underlying surface. For verification, we approximate the signed distance function to a set of 2D points using three methods: distance to closest point, distance to tangent plane at closest point and IMLS function. Figure 3.4 shows the estimated signed distance functions. As we can see, when the value of \( \sigma \) is selected appropriately, IMLS provides a smooth signed distance approximate. When the value of \( \sigma \) is too small, then the IMLS result becomes similar to that of the point-to-plane method. Figure 3.5 displays the reconstructed surfaces by extracting the zero level set from the corresponding signed distance function estimations. As can be seen in this figure, the surface implied by the IMLS function is a regression based on the sample points, whereas the functions based on point-to-point and point-to-plane distances in effect perform only a piecewise linear interpolation. Thanks to this regression property of IMLS function, SDR can deal with noisy point clouds in which the points do not exactly lie on the surface.

Thus, our proposed method is more general than the ICP method with the introduction of the smoothing parameter \( \sigma \). By varying the value of \( \sigma \), the behavior of SDR can be controlled and tuned. One advantage of SDR is that a point is matched to a location on the surface, which might not coincide with any point in the destination set. Thus, the method can deal with two point sets with different sampling density.

### 3.6.2 A Challenging 2D Toy Example

To gain a deeper insight into the behavior of the method, flatland simulation is performed. In this section, a challenging case will be used to demonstrate the advantage of SDR with respect to ICP methods. Figure 3.6 shows two
Figure 3.4: Input point cloud and the estimated signed distance function with different methods. IMLS method with an appropriate value of the smoothing parameter $\sigma$ provides a smooth estimate.
Figure 3.5: The surfaces reconstructed by extracting the zero level set from the estimated signed distance function. With appropriate value of $\sigma$, IMLS produces a smooth surface that nicely interpolate between the points.

Figure 3.7 shows the registration results with different methods. For our proposed methods, SDR and IMLS-ICP, we perform a simple tuning of the smoothing parameter $\sigma$: we start with a large value of $\sigma$ and reducing the value as the point sets move closer together.

As anticipated, both closest-point ICP and ICP with projective association are trapped in a local minimum and fail to converge. Despite both SDR and IMLS-ICP converge to the optimal solution, SDR converges after only 8 iterations. Figure 3.8 plots the residual error (with respect to the ground truth) versus iteration for all methods. It can be seen that closest-point ICP and projective ICP are trapped in a local minimum.
Figure 3.6: Input of the simulation. The two point sets are displaced significantly with respect to each other. The repetitive structure makes it hard to correctly register two point sets.

A look into the IMLS function at each iteration provides an insight into the behavior of the proposed method. In Figure 3.9, the IMLS function at iterations 1, 2 and 3 is shown for illustration. At the beginning, the value of $\sigma$ is large and the function in essence tries to fit a plane through the whole point set. Thus, at the beginning, the whole target points set collectively pulls the source points set closer. Gradually, as $\sigma$ decreases, the IMLS function refines to a smooth regression surface around the point set and each source point is only affected by local forces that pull it to the correct location.

3.7 Application in 3D Reconstruction

In 3D reconstruction, we need to register several point clouds to a common global coordinate system, merge them together and recover the 3D surface. Since registering frame to frame will lead to drifting problem due to the accumulation of registration error, a better way is registering each new frame onto a common model and updating the model after each frame. Our pipeline for registration and fusion of multiple depth images for 3D reconstruction is shown on Figure 3.10. First, each new depth frame is preprocessed to reduce
Figure 3.7: Registration results of different methods versus the ground truth. Closest-point ICP and projective ICP are trapped in a local minimum solution whereas SDR and IMLS-ICP converge to the optimal solution.

Details of the representation and updating of the model are provided in the following parts.

3.7.1 Model Representation

To represent the model, we sample the IMLS function and its gradient on a 3D grid within a bounding box around the object. This representation
Figure 3.8: Residual errors versus iterations for all algorithms. Closest-point ICP and Projective ICP fluctuate around a local minimum while SDR and IMLS-ICP quickly converge to the global minimum. Note that SDR only takes 8 iterations to achieve the solution.

(a) 1<sup>st</sup> iteration  (b) 2<sup>nd</sup> iteration  (c) 3<sup>rd</sup> iteration

Figure 3.9: The IMLS function at 1<sup>st</sup>, 2<sup>nd</sup> and 3<sup>rd</sup> iteration. The function gradually moves toward the point clouds as the value of $\sigma$ decreases.

allow us to quickly register new point set onto the model, since we can use interpolation to obtain the value of the signed distance function and gradient at any point within the bounding box. Given the current model constructed from the first $k$ frames, we can register the point cloud from $k + 1^{th}$ frame to the model using Algorithm 7 where in step 2(b) we compute the values of $f$ and gradient by interpolation on the volumetric grid. Note that our model is different from the model used in KinectFusion [50, 51], which is the Volumetric Range Image Processing (VRIP) method proposed in [52].
Figure 3.10: Our 3D reconstruction pipeline.
3.7.2 Model Update

With IMLS method, the value of the function can be quickly updated with new data points. Consider the current function $f$ in Equation 3.2, which is computed from $N$ data points. Assume that we need to update the function to incorporate $M$ new points. Then the updated function can be computed as:

$$f_{\text{updated}}(x) = \frac{w(x)f(x) + \sum_{j=1}^{M} n_j^T (x - p_j) \phi_j(x)}{w(x) + \sum_{j=1}^{M} \phi_j(x)}$$  \hspace{1cm} (3.31)

where $w(x) = \sum_{i=1}^{N} \phi_i(x)$.

Let $\nabla_x f(x)$ be the current gradient of $f$, computed from the first $N$ points. Let $\nabla_x f^\dagger(x)$ be the gradient computed from $M$ new points, using Equation 3.4. Since the new point set is registered to the model, we approximate $f^\dagger(x) \approx f(x)$, where $f^\dagger(x)$ is computed from the $M$ new point using Equation 3.2. The updated gradient can be computed as

$$\nabla_x f_{\text{updated}}(x) \approx \frac{w(x)\nabla_x f(x) + w^\dagger(x)\nabla_x f^\dagger(x)}{w(x) + w^\dagger(x)}$$  \hspace{1cm} (3.32)

Note that each new point only affects the value of the function a small neighborhood surrounding it, depending on the smoothing parameter $\sigma$. For that reason, with the volumetric model described above, we only need to update a small number of grid voxels when we integrate a new point onto current model.

3.8 Experimental Results

In this section, the experimental results are presented to evaluate the proposed method. First, we present the metric used for quantitative evaluation of different methods and the data that we used for the evaluations. Then, we show the results of registration between two point clouds. Finally, we provide the results of registering a sequence of depth images together.
3.8.1 Evaluation Metric

For quantitative comparison between methods, we will evaluate the deviation between the transformation obtained by registration versus the ground truth transformation.

Rotational Error

We evaluate rotational error by comparing the quaternions representing the resulting rotation and the ground truth. Let \( \hat{\mathbf{q}} \) and \( \mathbf{q}^* \) be the quaternions representing the estimated and the ground truth rotation, respectively. The rotational error is computed using the following metric [68]. The range of this metric is \([0, 1]\).

\[
e_{\text{rot}} = 1 - |\hat{\mathbf{q}} \cdot \mathbf{q}^*|.
\] (3.33)

Translational Error

The translational error is computed as the norm of the difference between the estimated translation \( \hat{\mathbf{t}} \) and the ground truth \( \mathbf{t}^* \)

\[
e_{\text{trans}} = \|\hat{\mathbf{t}} - \mathbf{t}^*\|.
\] (3.34)

Evaluation of Depth Sequence Registration

In registration of a synthetic depth sequence, we have the ground truth transformation between each frame and the first frame. We will evaluate the frame-to-frame rotational and translational errors at each frame. Consider the registration of the \(i^{th}\) frame to the global coordinate system. Let \(\mathbf{P}_i\) and \(\mathbf{Q}_i\) be the transformation obtained by registration and the ground truth transformation, respectively. The frame-to-frame transformation that registers the \(i^{th}\) to the \((i - 1)^{th}\) frame is computed as

\[
\mathbf{T}_{i-1,i} = \mathbf{P}_{i-1}^{-1}\mathbf{P}_i.
\] (3.35)

We also obtain the ground truth frame-to-frame transformation as

\[
\mathbf{T}_{i-1,i}^{(gt)} = \mathbf{Q}_{i-1}^{-1}\mathbf{Q}_i.
\] (3.36)
The rotational and the translational components are extracted from these frame-to-frame transformations and evaluated using the metrics presented above.

### 3.8.2 Test Data

In our experiments with synthetic data, we used several 3D models as shown in Figure 3.11. The 3D models of the bunny and the Buddha statue were obtained from the Stanford 3D Scanning Repository. The head model was obtained from the McGuire Graphics Data. For experiments that involve depth data, we generate the depth maps by projecting these models onto a virtual camera. The resolution of our synthetic depth map is 200 × 200. One of our synthetic depth maps is shown in Figure 3.12.

![Figure 3.11: The 3D high-quality models that we used to generate our synthetic test data.](image)

With synthetic data, we have ground truth information for quantitative evaluation of the registration performance.

### 3.8.3 Registration of Two 3D Point Clouds

This section evaluates the performance of registration methods to register two 3D point clouds. We perform these experiments on synthetic data for quantitative evaluation, using the metrics presented in Section 3.8.1. In this
Figure 3.12: An example synthetic depth map that we generate for the experiments

experiment, we transform a 3D point cloud using a random transformation. The transformed point cloud is used as the source set and is registered to the original point cloud. The transformations obtained with different methods will be compared with the ground truth transformation. To evaluate the robustness with missing data, we randomly keep only a portion of the source points. For each percent of source points that we keep, we run 100 tests, each with randomly kept source points and transformation. The transformation includes a random translation with maximum amount of 10 percent of the object dimension along each direction and a random rotation between $-45$ and $45$ degrees along a random axis.

The methods that we evaluated in this experiment are:

- Closest-point ICP with point-to-plane distance.
- Our proposed SDR methods.
- Our proposed IMLS-ICP variant.

Figure 3.13 compares the average rotational error and translational error with respect to the level of source points retained. As we can see in the figure, SDR achieves a smaller average error in most cases. The performances of IMLS-ICP and ICP are very similar in most cases.
3.8.4 Registration of a Depth Sequence

Figure 3.14 compares the rotational and translational frame-to-frame error when registering a sequence of depth images. In this experiment, we compare our method with closest-point ICP and ICP with projective association (Projection ICP). In our method, we register the depth map onto a common global model, which is a 3D grid of samples of the IMLS function. The input is a synthetic depth map with rotation angle about 3.6 degrees between consecutive frames. As we can see, all methods exhibit the accumulation of errors over time.

Figure 3.15 shows the output of our 3D reconstruction pipeline with the head model. The results were generated by registering and fusing 100 frames with a rotation angle of 1 degree between consecutive frames. There are some artifacts in our results near the shoulder of the model due to estimation error of normal vectors. This is because that region is nearly perpendicular to the camera viewing direction. One remedy for this problem is to filter pixels with normal nearly orthogonal to viewing direction when we update the model.

3.9 Concluding Remarks

In this chapter, we proposed the Signed Distance Registration (SDR) algorithm for registration of 3D point clouds. The key idea of SDR is that the source point set is registered to an approximation of the underlying surface based on the target point set. To approximate the surface, we borrow the IMLS method to estimate the signed distance function to the surface. We showed that cost function of ICP with point-to-plane metric in effect approximates the surface by linear interpolation, whereas IMLS performs a regression on the point cloud. With SDR, the behavior of the algorithm can be fine-tuned by varying the value of the smoothing parameter $\sigma$, as demonstrated in a 2D simulation. The incorporation of SDR into a complete 3D reconstruction pipeline was also introduced and experimented on 3D data. In the next step, we will perform deeper theoretical analysis on robustness and performance guarantees of the algorithm, as well as an efficient implementation for practical applications.
Figure 3.13: Average rotational and translational errors of different algorithms when registering two 3D point clouds. The number of source points is varied from 10% to 100% of the number of points in the original set. The errors are the average errors of 100 tests, each with randomly selected source points and transformation.
Figure 3.14: Frame-to-frame rotational and translational errors for the first 20 frames when registering a sequence of depth images together.
Figure 3.15: The two views of the reconstructed head model from 100 frames using our method. The artifacts around the shoulders are due to estimation error of normal vectors in the region orthogonal to the viewing direction.
Chapter 4

A Light-Field and Depth Camera System and Applications

4.1 Introduction

A light-field or plenoptic camera has the ability to capture the full 4D spatio-angular light-field of the scene. From a captured light-field, it is possible to synthesize images focused at different distances or images at slightly different perspectives. Recent researches on light-field cameras include calibrating the camera [69, 70], improving the quality and the resolution of refocused images [71, 72] and efficient methods for refocusing [73, 74].

The ability to record the 4D light-field is interesting and has a great potential for many novel applications. Light-field data is a nice supplement to the depth data provided by a depth camera. In theory, it is possible to estimate a depth map of the scene from the light-field, by using a multiview approach. Indeed, the light-field camera was originally designed for this purpose [75]. Recently, several researches have tried to achieve this goal, such as [71, 76, 77, 78]. However, the quality of the estimated depth maps is still not satisfactory for applications beyond refocusing. One reason is that with the microlens-based light-field cameras, the baseline is too small for stereo approach. Moreover, there is a trade-off between angular and spatial resolution in current cameras.

In this research we suggest combining the advantages of depth and light-field cameras to build a hybrid system that can exploit these complementary data. Our setup consists of a light-field camera and a depth camera rigidly attached together. In this chapter, we will present in detail the calibration process of the system and demonstrate it to refocus on an arbitrary surface of the scene. For this application, the inputs are a raw light-field image and a depth map of the scene that contains a textured surface. The result is an image in which all texture on the identified surface is in focus. Beyond the
refocusing application, we are interested in using our system to recover more complex properties of a surface, such as the surface light-field \([72]\).

The main contributions of this work are:

- A novel system that combines the advantages of depth camera and light-field camera for geometric and radiometric acquisition of objects.
- A detailed calibration process to take advantage of our system.
- A method to create a refocused image in which all details on an arbitrary surface is in focus.

This chapter is organized as follows. First, we review related work on the subject in Section 4.2. Then, Section 4.3 presents background material, especially the image formation model of microlens-based light-field camera. We provide a detailed description of the calibration process in Section 4.4. After that, we present the application of the proposed system to refocus on an arbitrary surface in Section 4.5. Section 4.6 provides a sketch of potential applications of our system to capture reflectance property of the surface. The experimental results and comparisons are given in Section 4.7. Finally, Section 4.8 will conclude this chapter.

### 4.2 Related Work

#### 4.2.1 Light-field

The concept of light-field was initially introduced in computer graphics for image-based rendering, where one would like to generate new views of a scene from existing views without geometry information. A light-field is a collection of radiance at points in space following different directions. If the region is free of obstacles, we can assume that the radiance is invariant along a ray. Levoy and Hanrahan \([80]\) and Gortler et al. \([81]\) independently proposed a 4D parameterization of light-field by the intersections of light rays with two parallel planes. This representation has been applied in many following researches. Liang et al. \([82]\) applied light transport and the 4D light-field representation to create a unified framework for modeling image formation. A variational analysis of light-field was presented in \([76]\) for estimation of
disparity maps. The application of light-field for digital image refocusing was proposed by Ng in [74]. In that work, the author introduced a frequency domain method to generate refocused image from light-field.

4.2.2 Light-Field Capture

The first camera that can capture a light-field directly, called a plenoptic camera, was proposed by Adelson and Wang [75] for capturing depth information with a single lens in one exposure. The authors suggested to place an array of tiny lenses or *microlens*, which they called a lenticular array, at the image plane to capture multiple images at slightly different viewpoints. Those images were used to estimate a depth map based on a multiview stereo approach. This design was improved by Ng et al. [83] to build the first hand-held light-field camera, which is later commercialized as the Lytro camera. In [84], the authors captured the light field using an array of spherical mirrors and performed digital refocusing from the captured data by using their proposed “axial-cone model”. A depth map of the scene is estimated with plane sweeping and is used to create an all-in-focus image. However, the proposed model is only applied for catadioptric systems with rotationally symmetric mirror. Georgiev et al. [85, 86] developed a new version of microlens-based light-field camera, which they called Plenoptic 2.0 camera, in which the microlens focus on the image plane of the main lens.

4.2.3 Calibration of Microlens-Based Light-Field Camera

The calibration of a light-field camera is complicated due to the addition of the microlens array. The first attempt to calibrate such a camera was presented by Dansareau et al. [70]. In that work, the authors developed a complete MATLAB toolbox to calibrate and rectify lenselet-based light-field camera. The authors decode the raw light-field image by estimating the locations of the center of the image under each microlens (the microlens image) and aligning these images to a rectangular grid. Finally, the raw image is sliced into the 4D light-field representation. A model with 12 intrinsic parameters is developed by modeling the main lens as thin lens and the microlens array as an array of pinholes. The calibration is done by minimizing
an error metric, named the “ray reprojection error”, between the corners of a checkerboard and the reprojected ray from each extracted observation in the raw image. The output of the calibration is a $5 \times 5$ matrix to project pixel coordinates to incoming light rays. Another calibration method for a light-field camera was proposed in [69] with a similar model but with only 6 parameters. The calibration is performed by extracting line features from the raw image without the construction of the 4D light-field representation.

4.2.4 Light-Field Rendering and Applications

Bishop and Favaro [71] introduced an image formation model for a light-field camera based on a spatially varying point spread function (PSF) and analyzed the aliasing of views. They proposed a Bayesian-based method to estimate depth and superresolve light-field data. Cho et al. [72] proposed a learning based approach to reconstruct light-field image. An efficient splatting-based method for refocusing from light-field is introduced by Fiss et al. in [73]. In that work, each pixel in the raw light-field image is projected to a ray, based on the estimated locations of the microlens centers, and splatted to the output image with a depth-adaptive splatting kernel. They also proposed an improved way to decode the raw light-field data, which estimates the microlens centers using both data image and the white image. Sabater et al. [87] proposed to demultiplex the raw image without demosaicing to avoid view cross-talk. They also proposed a new block-matching method to estimate the disparities from the light-field. Tao et al. proposed a method for estimating a depth map from light-field cameras by exploiting both defocus and correspondence information in [77]. An improved version for glossy surface was proposed in [78] in which multiple views in the light-field are exploited to separate the specular and the diffuse reflection. A detailed analysis of a microlens-based light-field camera using light transport was recently proposed by Liang and Ramamoorthi in [88].
4.3 Background

4.3.1 Projection Model of Microlens-Based Light-Field Camera

Our model for the light-field camera is similar to [70] and [69]. The main lens is modeled as a thin lens whereas each microlens is modeled as a pinhole. We denote $F$ as the focal length of the main lens, $L$ and $l$ as the main lens - microlens array and microlens array - sensor distance, respectively. Let $O$ be the center of the main lens. Figure 4.1 shows a 2D view of the camera model.

Consider a point $P$ at distance $D$ in front of the main lens. The image of $P$ by the main lens is $P'$ at distance $D'$ from the main lens. Using thin lens equation, $D'$ is computed as

\[ \frac{1}{D} + \frac{1}{D'} = \frac{1}{F} \]

\[ \Rightarrow D' = \frac{FD}{D - F}. \tag{4.1} \]

The distance from $P'$ to the microlens array is thus equal to $L - D'$. The point $P'$ is then projected on the sensor by the microlens array.

Consider the $k^{th}$ microlens in the array on Figure 4.1. This microlens projects the center of the main lens $O$ and the point $P'$ to the location
\( x_{ck} = [x_{ck}, y_{ck}]^T \) and \( x = [x, y]^T \) on the raw image, respectively. For all points in the scene, the \( k^{th} \) microlens would form a local image on the sensor, which is called the \( k^{th} \) microlens image with center \( x_{ck} \). The local coordinate of a pixel \( x \) in the \( k^{th} \) microlens image is defined as

\[
\begin{bmatrix}
u \\
v
\end{bmatrix} = 
\begin{bmatrix}
x - x_{ck} \\
y - y_{ck}
\end{bmatrix}
\]  

(4.3)

The output intensity of the raw image at each pixel is the integrated intensity of all rays that arrive on the pixel area. In practice, the sensitivity profile of the sensor is angle dependent. Thus, each incoming ray is weighted based on the angle between its direction and the surface normal of the sensor. A detailed image formation model that considers non-uniform sensitivity profile of the sensor is recently presented by Liang and Ramamoorthi in [88].

4.3.2 The center-view sub-aperture synthetic camera and its projection model

A sub-aperture image is defined as the collection of locations on the sensor which are the projections of the same point on the main lens, as shown on Figure 4.2.

![Figure 4.2: The formation of a sub-aperture image. The pixels having the same local coordinate in their corresponding microlens image are the projections of the same point on the main lens.](image)

In this figure, the point \( M \) at location \( d \) on the main lens is projected to locations \( x_1 \) and \( x_2 \) by microlens 1 and 2, respectively. Similarly, the
center $O$ is projected to locations $x_{c_1}$ and $x_{c_2}$. The local coordinates of the projections are $u_1$ and $u_2$. Due to the similarity between triangles, it is easy to see that $u_1 = u_2$:

$$u_1 = u_2 = -\frac{l}{d}.$$  \hspace{1cm} (4.4)

Hence, a sub-aperture image can be synthesized from the raw light-field image by extracting appropriate pixels, given the centers of the microlens image.

In our method, we are interested in the center-view sub-aperture image. This is the sub-aperture image formed by the rays going through the center $O$ of the main lens. This image can be thought of as a projection of the scene by a synthetic camera, which we model by a normal pinhole camera with distortion, similar to the model in [89].

We attach a world coordinate system $\{S\}$, which we will call the sub-aperture camera’s coordinate system, to this synthetic center-view sub-aperture camera and we will use superscript $(s)$ to denote the coordinates of a point in this coordinate system. The mapping from a point $P^{(s)} = [X^{(s)}, Y^{(s)}, Z^{(s)}]^T$ to a pixel $p^{(s)} = [u^{(s)}, v^{(s)}]^T$ on the sub-aperture image is as follows. First, the point is normalized as

$$P_n^{(s)} = [X_n^{(s)}, Y_n^{(s)}]^T = \left[\frac{X^{(s)}}{Z^{(s)}}, \frac{Y^{(s)}}{Z^{(s)}}\right]^T.$$  \hspace{1cm} (4.5)

Then, distortion is applied as

$$P_g^{(s)} = \begin{bmatrix}
2k_3^{(s)}X_n^{(s)}Y_n^{(s)} + k_4^{(s)} \left[r^2 + 2(X_n^{(s)})^2\right] \\
k_3^{(s)}(r^2 + 2(Y_n^{(s)})^2) + 2k_4^{(s)}X_n^{(s)}Y_n^{(s)}
\end{bmatrix},$$  \hspace{1cm} (4.6)

$$P_k^{(s)} = (1 + k_1^{(s)}r^2 + k_2^{(s)}r^4 + k_5^{(s)}r^6)P_n^{(s)} + P_g^{(s)},$$  \hspace{1cm} (4.7)

where $r^2 = (X_n^{(s)})^2 + (Y_n^{(s)})^2$ and $k^{(s)} = [k_1^{(s)}, \ldots, k_5^{(s)}]^T$ is the vector of distortion coefficients for the camera. Finally, the projection is obtained by

$$p^{(s)} = K^{(s)}\begin{bmatrix} P_k^{(s)} \\ 1 \end{bmatrix},$$  \hspace{1cm} (4.8)

where $K^{(s)}$ is the intrinsic matrix of the camera.
4.3.3 Projection model of the depth camera

Similar to the synthetic sub-aperture camera above, we also model the depth camera by a pinhole camera with intrinsic matrix $K^{(d)}$ and vector of distortion coefficients $k^{(d)}$. We attach a world coordinate system $\{D\}$ to the depth camera and call it the depth camera’s coordinate system. The coordinates of a pixel in this coordinate system will be denoted with superscript $(d)$. The projection of a point $P^{(d)}$ to a pixel $p^{(d)}$ in the depth camera follows the Equations (4.5) - (4.8), using appropriate parameters of the depth camera.

4.3.4 Coordinate Transformation Between the Depth Camera and the Sub-Aperture Camera

Since the depth camera and the light-field camera are rigidly attached together, the coordinate transformation from $\{D\}$ to $\{S\}$ is a rigid transformation. This transformation is represented by a rotation matrix $R$ and a translation vector $t$. A point with $P^{(d)}$ in $\{D\}$ will be transformed to $\{S\}$ by

$$P^{(s)} = RP^{(d)} + t.$$  \hspace{1cm} (4.9)

4.4 System Setup and Calibration

4.4.1 System Setup

Our system consists of a Lytro camera and an Asus Xtion Pro Live RGB-D camera rigidly attached together, as shown on Figure 4.3. Only the depth map from the RGB-D is used in our method. The input of our method include a depth map and a raw light-field image.

4.4.2 System Calibration

To obtain the intrinsic parameters of the cameras and the relative transformation $\{R, t\}$, we need to calibrate the system in advance. Since the calibration result depends on the zoom and focus of the light-field camera, it
Figure 4.3: System setup. Our system consists of a Lytro camera and an Asus Xtion Pro Live rigidly attached together.

is necessary to keep these parameters fixed during the calibration and data capture.

We perform the calibration with a checkerboard pattern with known dimension. The infra-red intensity images (captured by the depth camera) and the raw light-field images of the checkerboard at different orientations are used as input for the calibration process. The calibration process includes intrinsic and extrinsic calibration. The former includes the calibration of the depth camera, the center-view sub-aperture synthetic camera and the light-field camera. The latter involves the depth camera and the sub-aperture synthetic camera. We describe each of these calibration processes in detail below.

Intrinsic Calibration

To calibrate the depth camera, we use the infra-red intensity images, which are registered to the depth maps. From this calibration, we obtain the intrinsic matrix $\mathbf{K}^{(d)}$ and the distortion vector $\mathbf{k}^{(d)}$ of the depth camera, which allows us to project each pixel in the depth map to a 3D point in $\{D\}$.

To calibrate the sub-aperture synthetic camera, we first estimate the locations of the microlens center, using a method similar to [73]. Based on
the estimated center locations, we generate the center-view sub-aperture image by interpolating the raw light-field images at the center locations. We perform the calibration on the extracted sub-aperture image to get the intrinsic matrix $K^{(s)}$ of the synthetic camera. This matrix is used to project 3D points in $\{S\}$ to a pixel in the sub-aperture image.

To calibrate the Lytro camera, we use the method proposed by Bok et al. [69], which extracts the light features from the raw light-field images of the checkerboard and solves a system of linear equations. Given the calibration result, we obtain the estimated focal length $F$ of the main lens and the estimated distance $L$ between the main lens and the microlens array. For Lytro camera, the distance between the microlens array and the sensor is included in the metadata, obtained with each captured image, and is approximately 25 micrometers.

Extrinsic Calibration

To obtain the relative transformation $\{R, t\}$ between the depth camera’s coordinate system $\{D\}$ and the sub-aperture camera’s coordinate system $\{S\}$ (as defined in Equation (4.9)), we use the pairs of infra-red images and the corresponding sub-aperture images and perform a stereo calibration between them. We need to ensure that the region captured by the light-field camera overlaps with that captured by the depth camera. When we select the corners of the checkerboard, we make sure that the origin in the infra-red image is the same as that in the sub-aperture image.

To verify the calibration result, we select an arbitrary infra-red image and the corresponding light-field from the dataset. We select some checkerboard corners on the infra-red image and using the depth map to re-project them onto the sub-aperture image. Figure 4.4 shows that the projected corners are correctly aligned with the corners in the sub-aperture image.

4.5 Refocusing on an Arbitrary Surface

In order to refocus on an arbitrary surface, it is essential to know the depth map of the captured scene. One approach is estimating the depth map from the sub-aperture images. The sub-aperture images contain the shifted views
of the scene and a multiview stereo method can be applied to estimate the depth map from them. The estimated depth map, however, is affected by the low resolution of those images ($328 \times 328$ for Lytro camera) and by the aliasing between them. In [73], the authors created refocused images at different depth values and estimated the depth at each ray by choosing the value that minimizes a photoconsistency criterion. However, this process is tedious and time consuming. On the other hand, current commodity depth cameras can capture a high-quality depth map at video rate and are very affordable. For that reason, we propose to combine light-field camera with commodity depth camera to achieve our goal.

To project the depth map to the raw light-field image, we need to calibrate our system in advance. The dimension of a raw Lytro image is $3280 \times 3280$
pixels and each microlens image has a radius of only 5 pixels. If we select one pixel from each microlens image and put them together, the resulting image will have a small dimension of only 328 × 328 pixels. Figure 4.5 shows a demosaiced raw light-field image extracted from the Lytro camera. If we zoom in the image, we can see clearly the microlens images and we notice that each microlens image only covers a very small part of the scene. Since we are interested in part of the scene on a smooth surface, we can safely assume that the pixels under a microlens image have the same depth value. Thus we only need to map the depth map to the center-view sub-aperture image (i.e. the image consisting of the center pixels of the microlens images). Hence, we do a calibration between the depth camera and a synthetic camera that maps the scene to the center-view sub-aperture image, using the stereo calibration functionality provided in the CalTech calibration toolbox.

Given the calibration, we obtain a depth value for each microlens image. We map that depth value to a distance between the microlens array and the conjugate image plane (inside the camera) and apply the splatting method similar to [73] to synthesize the output result. Note that our derived splatting formula is different from the one in [73]. To be able to map the depth to the conjugate image plane, we need to know the focal length of the main lens and the distance between the main lens and the microlens array. We obtain these parameters by calibrating the Lytro camera using the method

\[ \text{http://www.vision.caltech.edu/bouguetj/calib_doc/} \]
proposed in [69].

4.5.1 Proposed Method

Our proposed method can be divided into the following main steps. The whole processing pipeline is presented in Figure 4.6. Some intermediate results for an example input are shown on Figure 4.8.

Figure 4.6: Our proposed method for refocusing. The input of the process is a depth map captured by the depth camera and a raw light-field image taken by Lytro camera. The output is a refocused image in which all details on an arbitrary surface are put in focus.

Estimate Depth Value at Microlens Centers

Since the depth camera’s field of view is wider than that of the Lytro camera, an appropriate region of interest is selected from the depth map and is projected onto the sub-aperture image. The input depth map is preprocessed with a bilateral filter to reduce noise. Each pixel $p^{(d)}$ in the region is projected to a point $P^{(d)}$ in $\{D\}$ using the intrinsic matrix $K^{(d)}$ and the distortion vector $k^{(d)}$. After that, the point is brought into $\{S\}$ by Equation (4.9). Finally, the point is projected onto the sub-aperture image plane by the following Equations (4.5) - (4.8). The depth of the point, i.e. its distance to the main lens or $Z^{(s)}$, is splatted to neighboring pixels in the sub-aperture image plane.

The result of this step is a depth map with the same size as the sub-aperture image, as shown on Figure 4.8(c). A 3D view of the projected depth map is displayed on Figure 4.8(d). Each pixel of the map corresponds to a microlens and contains a depth value. This depth value is the distance between the main lens and the point on the object that have a light ray going through the microlens.
Transform Depth Map to Distance Map

Since each plane in front of the main lens (at distance larger than focal length) has a conjugate image plane behind the main lens (inside the camera) related by the thin lens equation (4.1), we can transform the depth map resulted from the previous step to a distance map from the microlens array. For each depth value $D$ we apply Equation (4.1) to compute the distance $D'$ from the main lens to the conjugate plane. After that, the distance from the conjugate plane to the microlens array is obtained as

$$d = L - D'. \quad (4.10)$$

Generate Refocused Image

As the last step, we apply a splatting method similar to [73] to generate the output image. We assume that the pixels in the same microlens image share a common distance value, given by the distance map. Figure 4.7 describes the projection of a point on the imaged surface (formed by the main lens) onto the sensor via $k^{th}$ microlens in a 2D slice. Without loss of generality, the origin of the coordinate systems are set at the intersection between the principal axis with the corresponding planes. Let $p'$ be the projection of image point $P'$ on the sensor via $k^{th}$ microlens. Denote $x$ as the coordinate of $p'$, $x_C$ is the center of corresponding microlens image and $u = x - x_C$ is the local coordinate $p'$. Let $d$ be the distance value corresponding to the microlens.

As shown in the figure,

$$O'P' = O'B - EB - P'E$$

$$= O'B - OA \frac{d + l}{L + l} - HG \frac{d}{l}, \quad (4.11)$$

where the second term is due to the similarity between triangles $OHA$ and $EHB$ and the last term is due to the similarity between $EO_kP'$ and $HO_kG$. Since $O'B = OA = x_C$ and $HG = u$, the coordinate of $P'$ in the plane
containing it and orthogonal to the optical axis can be computed as

\[ x_{Pr} = x_C - x_{C, d + l \over L + l} - u {d \over l} \]

where the last equation follows from \( x_C = x - u \).

Since \( L \gg l \) and \( L \gg d \), the last term is negligible and can be omitted. Thus, Equation (4.13) becomes

\[ x_{Pr} = x - \left( 1 + {d \over l} \right) u . \tag{4.15} \]

Note that Equation (4.15) is different from the equation derived in \( [73] \).

In 3D, the splatting location is obtained by

\[ q = x - (1 + {d \over l})u. \tag{4.16} \]

We set the dimension of the refocused image \( J \) to the same dimension of the raw image \( I \). For each pixel \( x_i = [x_i, y_i]^T \) on the raw image, we find the center of the corresponding microlens images and the local coordinate \( u_i = [u_i, v_i]^T \) and apply Equation (4.16) to find the splatting location \( q_i \) on the output image. We splat the value of the pixel, weighted by the intensity \( w_i \) of the corresponding pixel in the white image, to pixels around the computed
splatting location using a splatting kernel. In our implementation, we use a
tent kernel, defined as

\[ k(d) = \max \left( 1 - \frac{d}{r}, 0 \right), \]  

(4.17)

where \( r \) is the radius of the kernel.

The output image is calculated as

\[ J(x) = \frac{1}{W(x)} \sum_i I(x_i)k(\|x - q_i\|)w_i, \]  

(4.18)

where

\[ W(x) = \sum_i k(\|x - q_i\|)w_i. \]  

(4.19)

In the above equations, \( i \) is in the range of the raw image dimension and
the kernel will limit the summation to valid rays around the output pixel \( x \).
Figure 4.8(e) shows the final refocused result. We can see that all the texture
on the surface is put in focus.

### 4.5.2 Layered Refocusing

For comparison, we also perform refocusing by using a layered method. In
this method, we create refocused images at several distances, between the
minimum and the maximum distance in the distance map, and we fuse these
images together. Let \( d_{\text{min}} \) and \( d_{\text{max}} \) be the minimum and maximum distance.
We divide the range \([d_{\text{min}}, d_{\text{max}}]\) into \( N \) equal intervals of length \( \Delta_d = (d_{\text{max}} - d_{\text{min}})/N \) and generate refocused images at distances \( d_n \), where

\[ d_n = d_{\text{min}} + n\Delta_d, n = 0, 1, \ldots, N. \]  

(4.20)

For layer \( n \), we compute the output image \( J_n \) and a map \( M_n \) that indicates
pixels in focused at that layer, using the following equations. We use a soft
mask with Gaussian weight for discrepancy between ray’s distance and the
refocused distance.

\[ J_n(x) = \frac{1}{W(x)} \sum_i I(x_i)k(\|x - q_{n,i}\|)w_i, \]  

(4.21)
\[ M_n(x) = \frac{1}{W(x)} \sum_i e^{-\frac{(D(x_i)-d_n)^2}{2\sigma_d^2}} k(\|x - q_{n,i}\|)w_i, \] (4.22)

where \( D(x_i) \) is the distance value at pixel \( x_i \) and \( \sigma_d \) is a smoothing parameter. \( W(x) \) is computed using Equation (4.19) and the splatting location \( q_{n,i} \) is computed as

\[ q_{n,i} = x_i - (1 + d_n/l)u_i. \] (4.23)

The final result is obtained by merging the layers together, using the masks as weights,

\[ J_{layered}(x) = \frac{\sum_{n=0}^{N} J_n(x)M_n(x)}{\sum_{n=0}^{N} M_n(x)}. \] (4.24)

Figure 4.9 displays the refocused image at two different layers and their corresponding masks for an example.

### 4.6 Potential Applications of the Proposed System

#### 4.6.1 Capture Reflectance Property of Surfaces

When the rays in the light-field are originating from a surface, the light-field becomes a surface light-field [79]. Given the geometry of the surface and the surface light-field, then virtual images of the surface in novel viewpoint can be generated. If the surface is Lambertian, i.e. reflected light is independent of the exit angle, then only a RGB image and the geometry of the surface is enough for this goal. However, if the surface is not Lambertian, such as a surface made of shiny material, then a surface light-field is required to render the virtual view realistically under complex lighting condition. Note that given a surface light-field, we can only generate novel views of the surface under the same lighting condition. To generate images under new lighting conditions, the Bidirectional Reflectance Distribution Function (BRDF) of the surface material is required. The BRDF is a function of input and output directions, which map the incident irradiance of the surface to the output radiance [90].

There is great interest in recovering the reflectance property, in particular the BRDF, of a surface using photometric methods [91, 92, 93, 94, 95, 96, 97]. With photometric methods, a set of images of the surface, under different
viewing direction, are captured. Each pixel in the image is mapped to a surface location and a viewing direction. This is repeated for different light directions. Thus, the process is computationally involved.

Our system, which can capture both geometry and light-field, can help to greatly reduce the acquisition time. Each pair of depth and light-field image can be mapped to several viewing directions in a cone, which is equivalent to several captured photographs. Thus, we speculate that our proposed system will be useful for reflectance capturing, on which we will focus in future research.

4.7 Experimental Results and Discussions

We perform experiments with inputs at different levels of complexity, from a simple tilted plane to a curved surface with complex geometry. In these experiments, the raw light-field images are captured at zoom step 413 and focus step 1230 (as shown in the Lytro metadata file). We estimate the locations of the microlens centers from the white images at corresponding zoom and focus settings. The input raw light-field images are color corrected and white balanced, using the parameter included in the Lytro metadata file. For each experiment, we perform a direct refocusing with depth, using the algorithm in Section 4.5.1, and a layered refocusing with the algorithm in Section 4.5.2. We compare our result with the result of Lytro Desktop software. Since the outputs of the Lytro Desktop software have resolution 1080 \times 1080, we perform a bicubic upsampling to the same resolution of our method, which is 3280 \times 3280. We used a splatting radius of 6 pixels in our experiments. For layered refocusing, we used 10 layers.

To demonstrate that the depth map produced with our system has better quality than the depth map estimated from the light-field, Figure 4.10 displays the two depth maps for comparison. To generate the estimated depth map, we used the plane sweeping approach with photometric consistency criterion used by the authors in [73].

Figures 4.11 and 4.12 compare the output of our methods on one test case and Lytro result. We recommend that the readers view the figures in the electronic version of this thesis. As we can see from the results, direct refocusing with depth provides results as good as the layered refocusing but with
less processing time. The results confirm that our proposed method can create output images when an arbitrary surface is put in focus. In Appendix A, we provide high resolution images of these results for better comparison. We also provide results with a highly bent document and a complex surface.

4.8 Concluding Remarks

In this chapter, we propose a hybrid system that consists of a light-field camera and a depth camera for advanced applications that exploit both depth and light-field data. In particular, we demonstrate our system for digital refocusing on an arbitrary surface. The system is calibrated in advance using a checkerboard with known dimension. Given the calibration result, we can perform refocusing on an arbitrary curved surface in the scene. Moreover, given the depth map, it is possible to recover the geometry of the surface and trace back the rays to recover more complex reflectance properties of the surface.
Figure 4.8: Intermediate results of our processing pipeline.
Figure 4.9: Refocusing results at two different layers and their corresponding masks.
Figure 4.10: Comparison between the depth map captured by our system and the estimated depth map using plane sweeping. Our depth map has better quality of the estimated depth map. As we see, the estimated depth map is very noisy.
Figure 4.11: Refocusing results with our direct refocusing method and layered refocusing method, in comparison with Lytro result. The bottom part highlight the different between our result and Lytro result.

Figure 4.12: Refocusing results with our direct refocusing method and layered refocusing method, in comparison with Lytro result for another input.
Chapter 5

Conclusion

In this thesis, we focused on the problem of signal recovery on sparse and noisy data. In particular, we looked at three main problems.

In Chapter 2, we proposed a new tree-based method to recover signal with sparse-tree structure. We defined the sparse-tree structure and exploited this special structure in our proposed Tree-based Orthogonal Matching Pursuit (TOMP) algorithm. We also provided a detailed analysis of the performance guarantees and the complexity of the algorithm. The results showed that our algorithm can achieve a competitive performance with lower complexity.

In Chapter 3, we focused on 3D reconstruction using a commodity depth camera. In this work, we proposed the Signed Distance Registration (SDR), a novel algorithm for point cloud registration. We present a theoretical and empirical comparison and between our method and ICP. The integration of the proposed method into a complete 3D reconstruction pipeline was described in detail with a demonstration.

In Chapter 4, we proposed a novel system consisting of a light-field camera and a depth camera. We presented in detail the calibration process of the system and demonstrated our proposed system with application in digital refocusing on an arbitrary surface.

Our work still has much room for improvement. For the next steps, we will look in the following directions

- Deeper theoretical analysis on the performance and robustness of the SDR algorithm.

- Effective implementation of SDR for fast 3D reconstruction.

- Advanced practical applications using our proposed hybrid light-field and depth camera system.
Appendix A

Additional Refocused Results using Our Proposed System

Figures A.1 - A.5 provide the experimental results in higher resolution and bigger size. For each experiment, we will show the result using our proposed direct refocusing method, layered refocusing and Lytro result.

Figure A.1: High-quality images of the results presented in Figure 4.8 and Figure 4.9, together with two views generated from Lytro Desktop.
Figure A.2: High-quality images of the results presented in Figure 4.11 together with two views generated from Lytro Desktop.
Figure A.3: High-quality images of the results presented in Figure 4.12, together with two views generated from Lytro Desktop.
Figure A.4: Results with a highly bent document.
Figure A.5: Results with a surface with complex geometry.
References


